1

Implementation of the Modified Monte Carlo Technique Using Importance Sampling on the Block Oriented System Simulator

THESIS

John B. Bennett
Captain, USAF

AFIT/GE/ENG/90D-03

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

1

AFIT/GE/ENG/90D-03

Implementation of the Modified Monte Carlo Technique Using Importance Sampling on the Block Oriented System Simulator

THESIS

John B. Bennett
Captain, USAF

AFIT/GE/ENG/90D-03

AFIT/GE/ENG/90D-03

Implementation of the Modified Monte Carlo Technique Using Importance Sampling on the Block Oriented System Simulator

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

John B. Bennett, BSEE
Captain, USAF

December, 1990

Acknowledgments

I'd like to thank my committee members, Mr Marty Desimio and Capt Byron Welsh, for sacrificing many hours of their valuable time to help me in my attempt to make this thesis work. Thanks are also given to my advisor, Lt Col Norman, for permitting me to try to make the project completely successful up to the last possible moment.

Thanks to Dan Zambon for keeping the computer system going through hard times. A special mention is due to Pam Young, whose witticism kept me on my toes.

I'd like to especially thank my family for giving me the support and encouragement to attempt this thesis effort (and AFIT). To my wife, Sandra, and my children, Kimberly and Adam, I give my utmost love and appreciation for being who they are and for what they endured during the entire AFIT time. I will make up the time and attention they missed because of AFIT.

John B. Bennett

Table of Contents

Page

Acknowledgments ii

Table of Contents iii

List of Figures vi

Abstract viii

I. Introduction 1-1

1.1 Background 1-1

1.1.1 Computer Simulation Evaluation Method. . . 1-2

1.2 Problem 1-4

1.3 Scope 1-5

1.4 Approach/Methodology 1-5

1.5 Equipment 1-6

II. Communication System Simulation 2-1

2.1 Simulation Approach 2-2

2.1.1 Signal Generation. 2-2

2.1.2 Filtering of Signals. 2-4

2.1.3 Channel Nonlinearities. 2-5

2.1.4 Receiver Structures. 2-5

2.2 Monte Carlo Technique 2-7

2.3 Modified Monte Carlo Technique Using Importance Sampling 2-10

2.3.1 Biasing/Unbiasing Procedure Using Gaussian Functions. 2-13

Page

2.3.2 Estimation Error. 2-15

2.4 Simulation Software Packages 2-16

2.4.1 Modular Structure. 2-16

2.4.2 Simulation and Programming Language. . . . 2-17

2.4.3 Topological Configuration. 2-17

2.4.4 Model Library. 2-18

2.4.5 Time and Event Driven Simulation. 2-18

2.4.6 Postp' ɔcessor. 2-18

2.4.7 User Interface. 2-18

2.5 Block-Oriented Systems Simulator 2-19

III. Modeling of the System 3-1

3.1 BPSK System Decomposition 3-1

3.1.1 Data Generator. 3-1

3.1.2 BPSK Modulator. 3-3

3.1.3 Channel. 3-4

3.1.4 BPSK Demodulator. 3-5

3.1.5 Error Detector. 3-9

3.2 BPSK System 3-12

3.2.1 Determining BPSK System Internal Parameters. 3-12

3.2.2 Determining System BER. 3-14

3.2.3 BPSK SYSTEM Simulation Parameter Determination. 3-16

3.3 BPSK SYSTEM Verification 3-18

IV. Implementing Importance Sampling 4-1

4.1 Modified BPSK System Decomposition 4-1

4.1.1 Biased Noise Function. 4-1

Page

4.1.2 Weight Generator. 4-2

4.1.3 Importance Sampling Error Counter. 4-5

4.2 Modified BPSK System 4-7

4.3 MODIFIED BPSK SYSTEM Testing 4-7

V. Conclusions 5-1

5.1 Conventional Monte Carlo Technique 5-1

5.2 Modified Monte Carlo Technique Using Importance Sampling 5-1

5.2.1 CHANNEL Module. 5-2

5.2.2 WEIGHT FUNCTION GENERATOR Module. 5-2

5.2.3 IMPORTANCE SAMPLING ERROR COUNTER Module. 5-4

5.3 Probable Problem 5-4

5.4 Recommendations 5-5

Bibliography BIB-1

Vita VITA-1

List of Figures

Figure Page

2.1. Elementary Communication Channel 2-2

2.2. Error Probability 2-6

2.3. Confidence Bands When Observed Value is 10^{-k} 2-10

2.4. Sketch of Operations Performed with Conventional Sampling . . 2-11

2.5. Sketch of Operations Performed with Importance Sampling . . . 2-13

2.6. Operational Flow of a Simulation Package 2-17

2.7. BOSS Software Structure 2-20

3.1. RANDOM DATA Internal Modules 3-2

3.2. BPSK MOD Internal Modules 3-3

3.3. BPSK MODULATOR Internal Module 3-4

3.4. CHANNEL Internal Modules 3-5

3.5. PSK_DEMOD_AND ERROR_COUNTER Internal Modules . . . 3-6

3.6. PSK MATCHED_FILTER DEMODULATOR Internal Modules . 3-7

3.7. REAL BPSK MATCHED FILTER DEMODULATOR Internal Modules 3-8

3.8. BPSK DEMODULATOR Internal Modules 3 9

3.9. SIM STOPPER Internal Modules 3-10

3.10. REAL ERROR COUNTER Internal Modules 3-11

3.11. BPSK ERROR COUNTER Internal Modules 3-12

3.12. BPSK SYSTEM Internal Modules 3-13

3.13. Simulation Bandwidth of a Gaussian Random Variable 3-16

4.1. WEIGHT GENERATOR Internal Modules 4-3

4.2. WEIGHT FUNCTION GENERATOR Internal Modules 4-4

Figure Page

4.3. IMPORTANCE SAMPLING ERROR COUNTER Internal Modules 4-6

4.4. MODIFIED BPSK SYSTEM Internal Modules 4-8

AFIT/GE/ENG/90D-03

Abstract

The purpose of this ~~effort~~ was to implement the Modified Monte Carlo technique using Importance Sampling on the Block Oriented System Simulator (BOSS). In this ~~thesis,~~ computer simulation techniques of communications systems were reviewed. Next, conventional Monte Carlo techniques and Modified Monte Carlo techniques using Importance Sampling were reviewed. Models of Binary Phase Shift Keying (BPSK) systems using both Monte Carlo techniques were implemented and simulated. Reasons for the model using Importance Sampling not working correctly are postulated.

The Monte Carlo technique is a method of ensuring that an inherently infinite procedure, such as determining system bit error rate (BER), can be determined within an appropriate accuracy and a confidence range after a set number of samples. Conventional Monte Carlo requires 10^{k+1} samples be generated to determine a BER of 10^{-k}. This number of samples results in an estimated BER in the range of 0.5 to 2.0 of the true BER. The number of samples required using conventional Monte Carlo techniques can result in unacceptable simulation times for low probability events. Importance Sampling is a method of reducing the number of samples required to determine an estimated BER with the same accuracy and confidence as conventional Monte Carlo.

Implementation of the Modified Monte Carlo Technique Using Importance Sampling on the Block Oriented System Simulator

I. Introduction

1.1 Background

Since communications systems have become more complex, analytical approaches to evaluating these systems have become difficult, if not impossible(9:89). The need to use simplifying assumptions to analyze modern communication systems analytically often leads to an incorrect and misleading evaluation. Engineers have turned to simulators to evaluate communications system performance. However, as the performance of communication systems improve, the evaluation process becomes more difficult because of the increasing system complexity and more time consuming because more samples are required to be generated by the simulator. The time needed to run a simulation is directly related to the number of samples required by the simulation to determine the performance.

As the number of required samples increase, the simulation time increases to an unacceptable level. Techniques have been developed to permit a known number of system errors to determine the performance within an acceptable margin of error. As communications systems improve, this method of evaluating the performance becomes too time consuming. The need to further reduce the amount of time required to run a simulation led to the development of using probabilistic assumptions in the communications system model. These assumptions reduce the number of samples required to remain within the same margin of error, and thereby decrease the simulation time.

The bit error probability, or bit error rate (BER), in a digital communication system is an important measure of system performance(2:1916). The BER is a measure of how many bits are likely to be received in error divided by how many bits are received.

1.1.1 Computer Simulation Evaluation Method. With the availability of simulation packages, the use of computers to simulate and evaluate the performance of communications systems has become commonplace(3:153). The speed and ability of the computer to perform complex repetitive mathematics eliminates the need for using simplifying assumptions in the communications system model(8:126). Relieving the engineer of tedious mathematical calculations permits more time to be devoted to design, rather than number crunching(1:1).

The fundamental approach of computer simulation evaluation of communication systems is to consider the output of the simulation as a series of independent events. The independent events are the input signal and the communication systems response to that input signal. The model of the communication system is known and entered into the simulator. The simulator's task is to generate samples of the input signal and process those samples through the communication system model. The model usually consists of a transmitter, a receiver, and a medium to carry the signal from transmitter to receiver. At the medium stage, a noise signal is added to the desired signal. The noise signal corrupts the data signal and causes the received signal to be different from the transmitted signal. The processed samples out of the receiver are stored as the received, or output, signals of the communication system.

The received signals are compared to the known input signals and any differences are counted as errors. Dividing the number of errors by the total number of samples yields an estimate of the BER. As the number of samples increases to infinity, the estimated BER approaches the true BER. Since the time in observing an infinite number of trials would never end, the process of finding the true value of the BER is not possible. Therefore, the Monte Carlo technique was created to find an estimated BER accurate within an acceptable range.

1.1.1.1 Monte Carlo Technique. Monte Carlo techniques are used to reduce the number of required simulation samples, and thereby reduce the time needed to measure the BER. In the Monte Carlo technique of estimating the probability of bit error, enough samples are generated such that a degree of confidence can be assumed that the estimate of the BER is approaching the true BER. In general, a minimum of 10^{k+1} samples, where 10^{-k} is the true BER, must be observed before there is an acceptable degree of confidence in the observed value of the BER estimate(9:93)(3:157). The 10^{k+1} samples define a confidence interval where the estimated BER will range from one half to twice the actual BER. This interval is considered an acceptable uncertainty(3:157).

Random number generators in simulators generate the signals used for communication system models. The input signal for digital systems is random data where all possible signal values occur with an equal probability. The noise input used for communications systems is additive white gaussian noise (AWGN). The amplitude of the noise occurs according to the Gaussian probability density function (pdf). Samples of the Gaussian pdf are generated by a random number generator, added to the desired signal in the transmission medium, and processed through the communication system model by the simulator. A communication system with an expected BER of 10^{-6} requires 10^{7} samples to be processed before sufficient errors can be counted to arrive at an acceptable confidence level in the BER estimate. As an example, about 2 1/2 days are required to run a simulation of 10^{7} samples using the Block Oriented Systems Simulator (BOSS)(13:5-2).

The time required to evaluate the performance of a system requir g a large number of samples must be r luced. One method of reducing the simulation time is to modify the noise input signal to make errors occur more frequently. The data is then manipulated to remove the effect of the biased input. Using a modified probabilistic input noise signal is referred to as the Modified Monte Carlo technique using Importance Sampling.

1.1.1.2 Modified Monte Carlo Technique. Modified Monte Carlo simulation, using Importance Sampling, is a method of reducing the number of samples required to estimate the BER. Importance Sampling biases the Gaussian noise pdf to increase the rate at which errors occur. The simulator compares the signal input and signal output and decides if an error has occurred. The simulator then determines the weight of the noise during the symbol that caused the error to occur. The errors are counted using these weights. The effect of biasing the input pdf is removed by this weighting and an estimated BER is determined. Reducing the number of samples reduces the time required to run a simulation. Theoretically, using Importance Sampling can result in significant reduction in the number of required samples and still achieve the same level of confidence for a simulation using the conventional Monte Carlo technique.

1.2 Problem

In evaluating the performance of communication systems, a key parameter is the probability of bit error. Because of the complexity of current communication systems, computer simulation is used to evaluate the BER. The Monte Carlo technique is used to determine the BER within an acceptable range. The Modified Monte Carlo technique using Importance Sampling is one method to reduce the number of samples required to evaluate the BER within an acceptable range.

The Block Oriented Systems Simulator (BOSS) does not support any sample reduction method to reduce simulation time. This purpose of this thesis is to implement the Modified Monte Carlo technique using Importance Sampling on the BOSS simulator.

1.3 Scope

The Block Oriented Systems Simulator (BOSS) will be used to simulate the communication system and evaluate the BER using the conventional Monte Carlo technique and the Modified Monte Carlo technique with Importance Sampling. Importance Sampling will be the only type of sampling used in the Modified Monte Carlo simulation. Other sampling methods such as tail extrapolation, extreme-value or quasi-analytical will not be used.

The Gaussian pdf will be the only input noise type considered in this thesis. The principal simulation parameter to be changed in the all simulations will be the random number generator seeds. No other signal parameter biasing will be used. The performance factor of the communication system to be measured and compared is the BER.

All simulations will be performed using baseband signals to speed up the individual simulations. The communication system used will be simple for ease in understanding. The system will be set up using common modulators, demodulators, filters, and number generators.

1.4 Approach/Methodology

A digital communication system with a BER of about 10^{-5} will be modeled. A conventional Monte Carlo simulation of the communication system will be run using Gaussian amplitude distributed noise with a variance of one as the noise sequence to establish a baseline. Ten simulations will be made varying the seeds of the random number generators. From these runs a normalized error of the estimated BER will be determined.

The communication system will be modified to include the modules for using the Modified Monte Carlo technique. The variance of the Gaussian noise signal will be changed. As before, ten simulations will be made using different seeds for the random number generators and and the normalized error will be determined. The sample reduction will be determined that maintains the same level of uncertainty as the conventional Monte Carlo method.

1.5 Equipment

The equipment used in this thesis is that necessary to run the simulation:

- BOSS User's Manual
- BOSS Software Version: ST*AR 2.02
- DEC VAXstation 3

II. Communication System Simulation

The central issues in the design of communication systems are performance evaluations and tradeoff analysis(8:126). The use of computer-aided modeling or analysis to evaluate a communication system depends on factors such as system complexity and required accuracy (8:126). Except for some idealized and often over-simplified cases, it is extremely difficult to evaluate the performance of complex communication systems using analytical techniques alone(10:5). Analytical techniques are viable only under limited circumstances, and the network being modeled must often be unduly distorted to fit a model amenable to analytical solution(8:126). In fact, because of the distortions introduced to the system model, often one can wind up with the right solution to the wrong problem(8:126).

It has become increasingly common to use computer-aided techniques to estimate the performance of digital communication systems(3:153). Simulation is a powerful tool for evaluating the performance of communication systems(8:126). Simulation is often the only viable method to evaluate systems where the complexity is too involved and analytical techniques are too difficult(8:126). Some major advantages of analysis using simulation over analytical techniques are that simulations are more accurate and the engineer is freed from tedious number-crunching calculations(8:126).

This chapter is a summary of computer simulation techniques as these techniques apply to communication systems. The first section is an overview to modeling communications systems. The next two sections will describe the Monte Carlo method and the Modified Monte Carlo method using Importance Sampling as the evaluation techniques. Next, general features of simulation software packages are presented. Finally, the features of the Block Oriented Systems Simulator (BOSS), the simulator used in this thesis, are summarized.

2.1 *Simulation Approach*

Four major steps are fundamental to the simulation of any communication system. These steps include creating and storing the sampled representation of the signal, filtering the sampled signal, operating on the signal according to channel nonlinearities, and demodulating the signal to observe the effect of the channel(9:89). These steps will be discussed in terms of an elementary communication channel as shown in Figure 2.1.

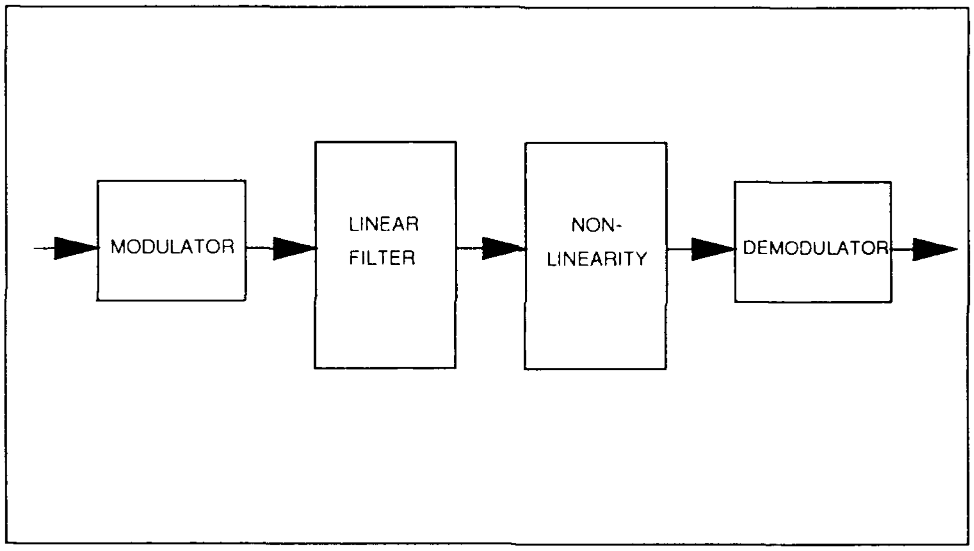

Figure 2.1. Elementary Communication Channel (9:90)

2.1.1 *Signal Generation.*

The first step of a simulation is to create a set of sampled signals. The sampled signals are representative of the real signals. In order to reduce the sampling rate and the simulation time, bandpass signals are normally represented by their low-pass equivalents (complex envelopes)(1:3). A *bandpass* signal, $x(t)$, may be represented by

$$x(t) = X_D(t)cos(2\pi f_c t) - X_Q(t)sin(2\pi f_c t) \qquad (2.1)$$

where

$$\begin{aligned} x(t) &= \mathit{time\,domain\,signal} \\ X_D(t) &= \mathit{direct\,component} \\ X_Q(t) &= \mathit{quadrature\,component} \\ f_c(t) &= \mathit{center\,frequency\,of\,the\,process} \end{aligned}$$

Using standard trigonometric identities, the bandpass signal can be equivalently represented as

$$x(t) = R(t)cos[2\pi f_c t + \Phi(t)] \tag{2.2}$$

where

$$\begin{aligned} R(t) &= [X_D^2(t) + X_Q^2(t)]^{1/2} \\ \Phi(t) &= arctan[\tfrac{X_Q(t)}{X_D(t)}] \end{aligned}$$

The *complex envelope* of *x(t)* is then

$$X(t) = R(t)\exp[j\Phi(t)] \tag{2.3}$$

This representation of the signal is known as the *complex low-pass* representation. Use of the complex representation significantly reduces simulation time because the center (carrier) frequency is not required in the simulation.

For a sampling frequency f_s, the complex envelope representation is(9:90)

$$x(k \cdot dt) = A(k \cdot dt)P^{1/2}\exp[j2\pi(\frac{f_o}{f_s})k + j\Phi(k \cdot dt)] \tag{2.4}$$

where

$A(k \cdot dt)$	*represents amplitude modulation*
$\Phi(k \cdot dt)$	*represents phase modulation*
$(\frac{f_c}{f_s})$	*is the relative carrier frequency*
P	*is the signal power*
$f_s = 1/dt$	*is the sampling frequency*
k	*is the sample index*

The main consideration in selecting the sampling frequency f_s is the total bandwidth required around the channel carrier frequency. In a digital communication system, the bandwidth is determined by the frequency spectrum of the transmitted symbol. Nyquist's Sampling Theorem states that only 2 samples per symbol are necessary to reconstruct the original signal. For simulation purposes, the sampling frequency should be an even integer between 8 and 16(9:90). If the sampling frequency is less than 8 samples per symbol, accuracy is lost; with more than 16 samples per symbol, excess simulation time is used without significant gain in accuracy(9:90).

2.1.2 Filtering of Signals. The next consideration in the simulator is the use of filters in the system. The primary purpose of system filters is to give an acceptable tradeoff between desired signal distortion and interference from adjacent channels(9:91). The simulator requires a flexible capability to model different gains/delays versus the frequency functions(9:91).

Efficient algorithms are necessary for a simulator because filtering tends to be the most CPU-intensive aspect of the simulation(9:91). There are three methods usually used to simulate filter actions: 1) convolve the sampled signal with a stored filter impulse response; 2) multiply the discrete Fourier transform (DFT) of the signal by the filter frequency transform function and take the inverse DFT to return the signal to the time domain representation; and 3) operate on the signal according to the difference equation governing the filter(9:91). Direct convolution, the first method, is rarely used because of the excessive CPU time required to perform the computations. The DFT approach is the most general and commonly used method to simulate the effects of filtering. However, for low order classical filters, such as the Butterworth or Chebyshev, the difference equation method is quicker than the DFT method.

2.1.3 Channel Nonlinearities. Analytical approaches become complex, if not impossible, to use when the system model contains nonlinearities. Simulation is often the only practical approach to perform system evaluation(9:92). Most simulations of devices with abrupt nonlinearities generate input/output tables to map input values to output values. These lookup tables are used when the simulator encounters abrupt changes to the input value and interpolation of the data points are used if necessary. For gradual nonlinearities in the simulation, power series representations of the input/output relationship are used(9:92).

2.1.4 Receiver Structures. The first module in the receiver structure is the receiver front-end filter. At this point in the model the remaining steps in the simulation include the addition of noise, demodulation, and quantitative performance evaluation(9:93). Performance evaluation would be measured by the signal to noise ratio for analog systems and the bit error rate for digital systems(9:93).

Two types of simulations are used depending on whether thermal noise, usually Gaussian, is added. The hybrid simulation/analysis approach is used if thermal noise is not added in the simulator. This approach uses simulation to obtain the statistics of any other sources of signal interference or distortion. The effect of thermal noise is then added to the statistics and an average error rate is calculated(9:94). In the direct simulation approach, thermal noise is added, demodulation is performed, and the performance of the overall system is evaluated. For digital systems, error counting is required, therefore making these simulations inherently time-consuming(9:94). The

Monte Carlo technique is a method of ensuring that simulating a certain number of symbols will yield the estimated BER within a range of accuracy and a level of confidence.

The demodulation and detection processes reduces the waveform to a number which is then compared to a threshold. The decision process can be described in relation to the probability density functions (pdf) $f_0(v;\tau)$ and $f_1(v;\tau)$, of the input voltage at the sampling instant τ. The detection decision is made relative to the input voltage at the sampling time τ given that a "one" or a "zero" was sent. Deciding a "one" was received when a "zero" was sent occurs when there is a large positive excursion of the receive voltage from the value of a "zero". If the excursion of the voltage is sufficiently large to exceed the threshold voltage V_T, an error will occur. Sample densities are shown in Figure 2.2. These densities may, in general, assume any shape and are not necessarily identical.

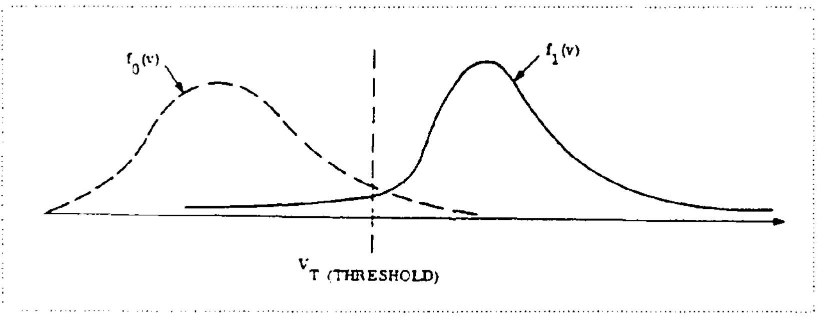

Figure 2.2. Error Probability(3:155)

The probability of error, given that a one was sent, is

$$\text{Prob[error/one]} \doteq p_1 = \int_{-\infty}^{V_T} f_1(v)dv \tag{2.5}$$

The probability of error, given that a zero was sent, is

$$\text{Prob[error/zero]} \doteq p_0 = \int_{V_T}^{-\infty} f_0(v)dv \tag{2.6}$$

The average probability is then

$$p = \pi_1 p_1 + \pi_0 p_0 \tag{2.7}$$

where

π_1	*is the* a priori *probability of the symbol "one"*
π_0	*is the* a priori *probability of the symbol "zero"*

2.2 Monte Carlo Technique

The Monte Carlo method is basically an error counting technique. In other words, the Monte Carlo technique estimates the probability of bit error by observing the number of error occurrences. Suppose that a "zero" was sent. Then the probability of error is

$$p_0 = \int_{V_T}^{\infty} f_0(v)dv \tag{2.8}$$

This may be rewritten as

$$p_0 = \int_{-\infty}^{\infty} h_0(v) f_0(v)dv \tag{2.9}$$

where h_0 is an error detector given by

$$h_0(v) = \begin{cases} 1, & v \geq V_T \\ 0, & v < V_T \end{cases}$$

Recognizing that Equation 2.9 expresses the expected value of h_0, the probability of error can be estimated with a sample mean of h_0. In other words, the probability of error

$$p_0 = E[h_0(v)] \tag{2.10}$$

can be estimated by counting the number of errors over N trials as given by

$$\hat{p}_0 = \frac{1}{N}\sum_{i=1}^{N} h_0(v_i) \tag{2.11}$$

where h_0 is an error detector, the summation is an error counter, $v_i = v(t_i)$, and t_i is the symbol-spaced instant at which the decision is made.

Equation 2.11 defines the Monte Carlo method(3:156). The probability of bit error is estimated by counting the number of error occurrences over a set of trials. Simplifying the above equation for η bit errors out of N bits observed defines an unbiased Monte Carlo estimate of the probability of bit error $\hat{p}$ as

$$\hat{p} = \eta/\mathrm{N} \tag{2.12}$$

where

η	*is the number of errors*
N	*is the total number of bits*

As the number of trials increases to infinity, the error probability estimate $\hat{p}$ approaches the true error probability p. The simulation time required for an infinite number of trials would never end and is therefore unacceptable. The *Monte Carlo* method was created in order to find a relatively accurate estimate of p.

In the Monte Carlo method of estimating the probability of error p , enough errors need to be counted to ensure that a degree of confidence can be assumed that the error probability estimate $\hat{p}$ is relatively close to the actual error probability p. The confidence level can be defined as(3:156)

$$P[h_2 \leq p \leq h_1] = (1 - \alpha) \tag{2.13}$$

where

$h_1 - h_2$	*is the confidence interval*
$(1 - \alpha)$	*is the confidence level*

In other words, $(1 - \alpha)$ is the probability that the true value of the BER p is bracketed by h_1 and h_2. Figure 2.3 shows the number of trials that needs to be observed before the estimate error rate lies within a desired confidence level of the actual error rate.

In general, a minimum of 10^{k+1} symbols should be observed before there is an acceptable degree of confidence in the error estimate $\hat{p}$ (3:157). This is equivalent to a vertical slice on the chart where $N = (10^{k+1})$. Going across the chart at the 90% confidence range shows that observing 10^{k+1} symbols defines a confidence interval where the estimated error probability $\hat{p}$ is in the range of $0.5p$ to $2p$. This interval is considered acceptable(3:157).

Figure 2.4 illustrates the operations performed by a Monte Carlo simulation. The input x to the processor is a random variable of a known probability density function, $f_X(x)$. Using a finite number of observations, we want to estimate the number of errors occurring in y. The sorting operation is deciding if an error has occurred. The output z is the number of errors.

However, use of the conventional Monte Carlo technique is limited to error probabilities in the region of 10^{-2} , 10^{-3} , and possibly 10^{-4} (9:93). Verifying lower

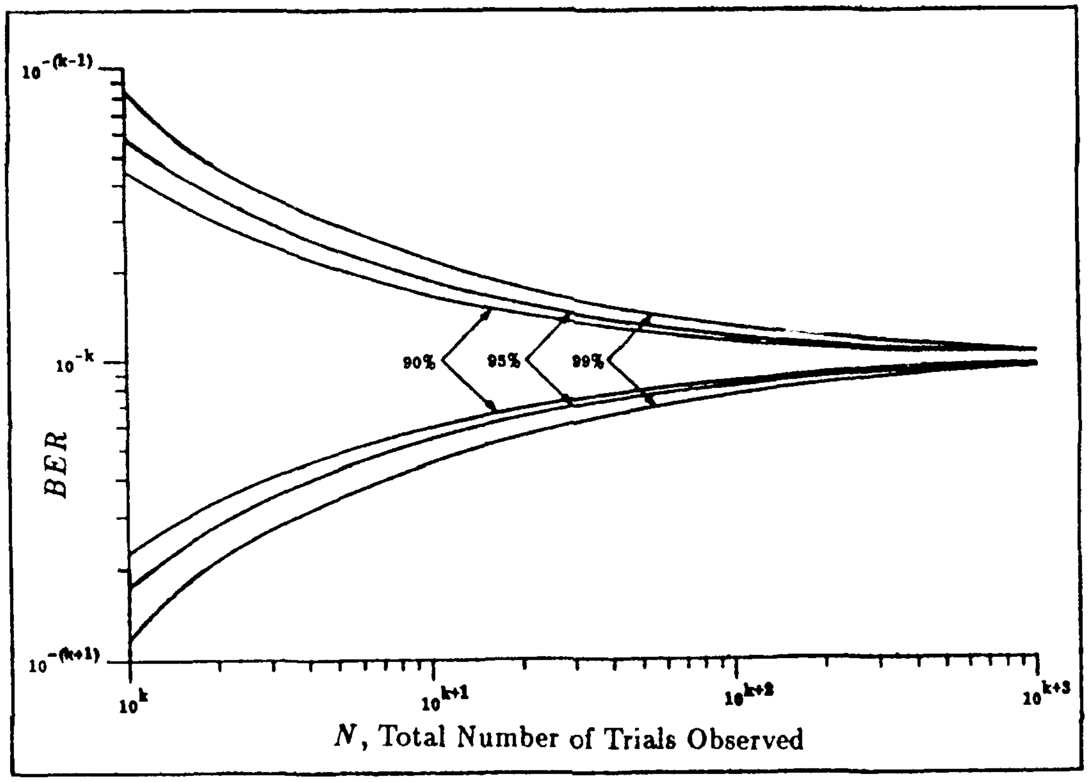

Figure 2.3. Confidence Bands When Observed Value is 10^{-k} (3:158)

error rates becomes extremely time consuming because of the number of samples the simulator is required to generate. A variation of the conventional Monte Carlo method using Importance Sampling has been developed to reduce the number of samples required, thereby reducing the simulation time.

2.3 Modified Monte Carlo Technique Using Importance Sampling

The large number of samples required to be generated for a small number of errors to be counted make the Monte Carlo method inefficient. The majority of the generated signals do not cause errors (the "important" event). If the noise is zero-mean Gaussian, for example, the most probable value is zero, and zero will

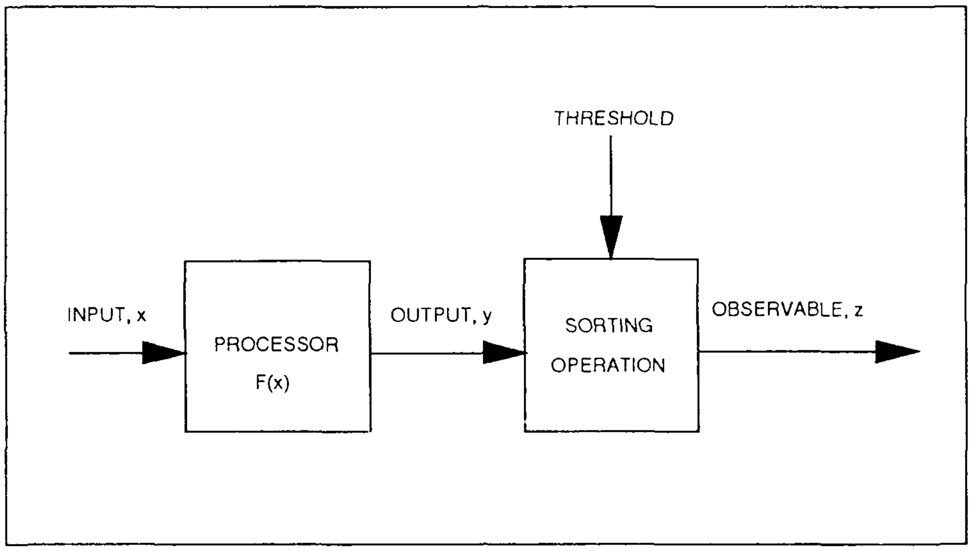

Figure 2.4. Sketch of Operations Performed with Conventional Sampling (6:17)

not generate an error. There is a certain range of values centered around the mean value that also do not contribute to error occurrences. The samples generated that do not induce errors are, in effect, wasted. Importance sampling is a method of making errors occur more frequently by a deliberate distortion, called biasing, of the statistics of the underlying noise process that causes errors to occur(3:158).

The probability of error given a "zero" was sent, as shown in Equation 2.9, can be rewritten as

$$p_0 = \int_{-\infty}^{\infty} h_0(v) \frac{f_0(v)}{f_0^*(v)} f_0^*(v) dv \tag{2.14}$$

where $f_0^*(v)$ is the *biased* density function. Noise samples from $f_0^*(v)$ will cause the errors to occur more frequently because, as shown later, f_0^* will be formed from f_0 in such a way as to increase the noise power(3:159).

The ratio

$$w(v) = f_0(v)/f_0^*(v) \tag{2.15}$$

is called the *weight*, or "unbiaser", of the function and the inverse of $w(v)$

$$B(v) = w^{-1}(v) = f_0^*(v)/f_0(v) \tag{2.16}$$

is called the *bias* of the function.

Equation 2.14 can be rewritten as

$$\begin{aligned} p_0 &= \int_{-\infty}^{\infty} h_0(v)w(v)f_0^*(v)dv & (2.17) \\ &= \int_{-\infty}^{\infty} h_0^*(v)f_0^*(v)dv & (2.18) \end{aligned}$$

where $h_0^*(v)$ is an error detector, not now based on a count of one as in conventional Monte Carlo, but now based on the weight $w(v)$.

As shown in Section 2.2, Equation 2.18 may be written as

$$p_0 = E[h_0^*(v)] \tag{2.19}$$

As before, the estimate of this equation can be expressed as the sample mean

$$\begin{aligned} \hat{p}_{0*} &= \frac{1}{N_*}\sum_{i=1}^{N_*} h_0^*(v_i) & (2.20) \\ &= \frac{1}{N_*}\sum_{i=1}^{N_*} H_0(v_i)w(v_i) & (2.21) \end{aligned}$$

where $H_0(v)$ is identical to $h_0(v)$ but the changed symbol is a reminder that $H_0(v)$ is the error detector when the biasing function f_0^* is the governing density(3:159). The $w(v_i)$ sequence of *observed* values is interpreted as the *weighting* function given in Equation 2.15 evaluated at the sequence of observed voltages $v(t_i)$. Equation 2.21 is the conceptual basis for producing an estimate using the modified Monte Carlo technique(3:159).

Figure 2.5 illustrates the operations performed by the Modified Monte Carlo simulation using Importance Sampling. As before in Section 2.2, the input x to the processor is a random variable of a known probability density function $f_X(x)$. Using a finite number of observations, we want to estimate the number of errors occurring in y. The sorting operation decides if an error has occurred. However, the output z_m is now the *weighted* number of errors.

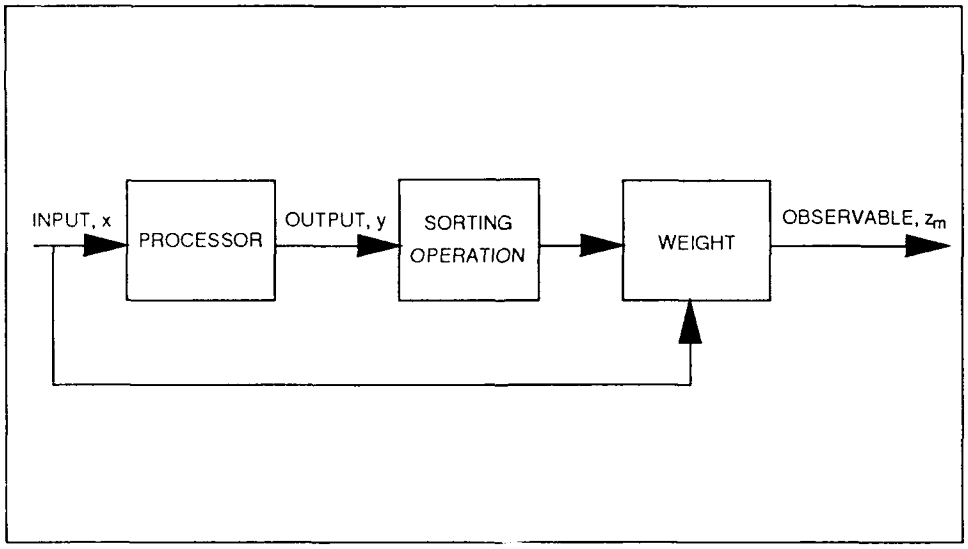

Figure 2.5. Sketch of Operations Performed with Importance Sampling (6:17)

2.3.1 Biasing/Unbiasing Procedure Using Gaussian Functions. Let X be a zero-mean Gaussian random variable. The zero mean Gaussian pdf with variance σ^2 is

$$f(x) = (1/\sqrt{2\pi}\sigma)\exp(-x^2/2\sigma^2) \tag{2.22}$$

For importance sampling, the Gaussian noise pdf is changed to cause errors to occur more frequently. The biasing function $B(x)$ is defined as

$$B(x) = \frac{f_X^*(x)}{f_X(x)} \tag{2.23}$$

where the variance σ^2 of $f(x)$ is unity and the variance σ_*^2 of $f^*(x)$ is greater than one so errors will occur more frequently. If $f_X^*(x)$ is used as the noise pdf rather than the original $f_X(x)$, then more samples will come from the region of interest (the tails of the pdf)(2:1918).

The biasing scheme suggested is (2:1920)

$$B(x) = \frac{c}{[f_X(x)]^\alpha} \tag{2.24}$$

where the constants c and α are chosen such that the new density $f_X^*(x)$ has unity area. This means that

$$\int_{-\infty}^{\infty} f_X^*(x)dx = \int_{-\infty}^{\infty} B(x)f(x)dx = 1 \tag{2.25}$$

The α is optimized for the BER of a particular system using(2:1920)

$$\alpha_{OPT} = \frac{-(2M+1) + \sqrt{(2M+1)^2 + 4T^2(1+T^2)}}{2T^2} \tag{2.26}$$

where M is the memory in the system and T is the threshold defined by $T = Q^{-1}(\hat{P}_B)$. This means that

$$c = \sqrt{\frac{1-\alpha}{(2\pi)^\alpha}} \tag{2.27}$$

The weighting function $w(x)$ is given by $f(x)/f^*(x)$ for a single sample point. From Equation 2.21, the BER is estimated by counting the errors and unbiasing each count by the ratio of the input densities. For the case where there is several samples per symbol and the input samples are independent, the *weight* of the noise that caused the error to occur is given by(3:160).

$$w(x_i) = \prod_{j=1}^{M} f_X(x_{i-j\Delta})/f_X^*(x_{i-j\Delta}) \tag{2.28}$$

where f_X and f_X^* are single dimensional (assumed stationary) densities, i is the symbol index, j is the sample index within the symbol, and M is the number of samples within the symbol.

2.3.2 Estimation Error. Associated with simulation is the error of the estimated value with respect to the calculated value. The number of samples required to estimate P_e by direct counting is(2:1922)

$$N > \frac{1}{\epsilon^2 P_e} \tag{2.29}$$

where ϵ is the normalized error of the estimated $\hat{P}_e$. Rewriting Equation 2.29, the normalized error is

$$\epsilon = \frac{\sigma_{\hat{P}_e}}{P_e} \tag{2.30}$$

where $\sigma_{\hat{P}_e}$ is the standard deviation of the estimate $\hat{P}_e$.

When computer simulation has been chosen to evaluate a communication system, it is important to choose a simulation software package that is both flexible and accurate.

2.4 Simulation Software Packages

Several software packages are now available for simulating communication systems. The features that make one simulation program a better choice for a particuar application are not clear-cut(1:2). However, it is important to understand the features that a communication simulation package should have, and a summary of desirable features as identified by Balaban and Shanmugan (1) are listed below.

2.4.1 Modular Structure.

To provide maximum flexibility, simulation software packages used for communication systems analysis and design should have a modular structure. Most simulation software packages in use now are made up of four major components: the model library, the system configurator, the simulation exercisor, and the postprocessor. The operational flow of a simulation package is shown in Figure 2.6.

The model library is used to store a set of functional blocks that can be accessed by the modeler to build a system to be simulated. The system configurator is used to configure the set of models selected from the model library in the desired topology. The individual block simulation parameters may be entered at the configuration level. The simulation exerciser generates the simulation sequence and stores histories of events or waveforms that occur at various points in the communication system model. Individual block simulation parameters may be entered and system level simulation parameters must be entered at the simulation exercisor level. At the conclusion of the simulation, the post processor outputs are used to verify if the design requirements and design constraints are met. Iterations of the simulation may be run by varying the parameters and using the postprocessor to compare the results.

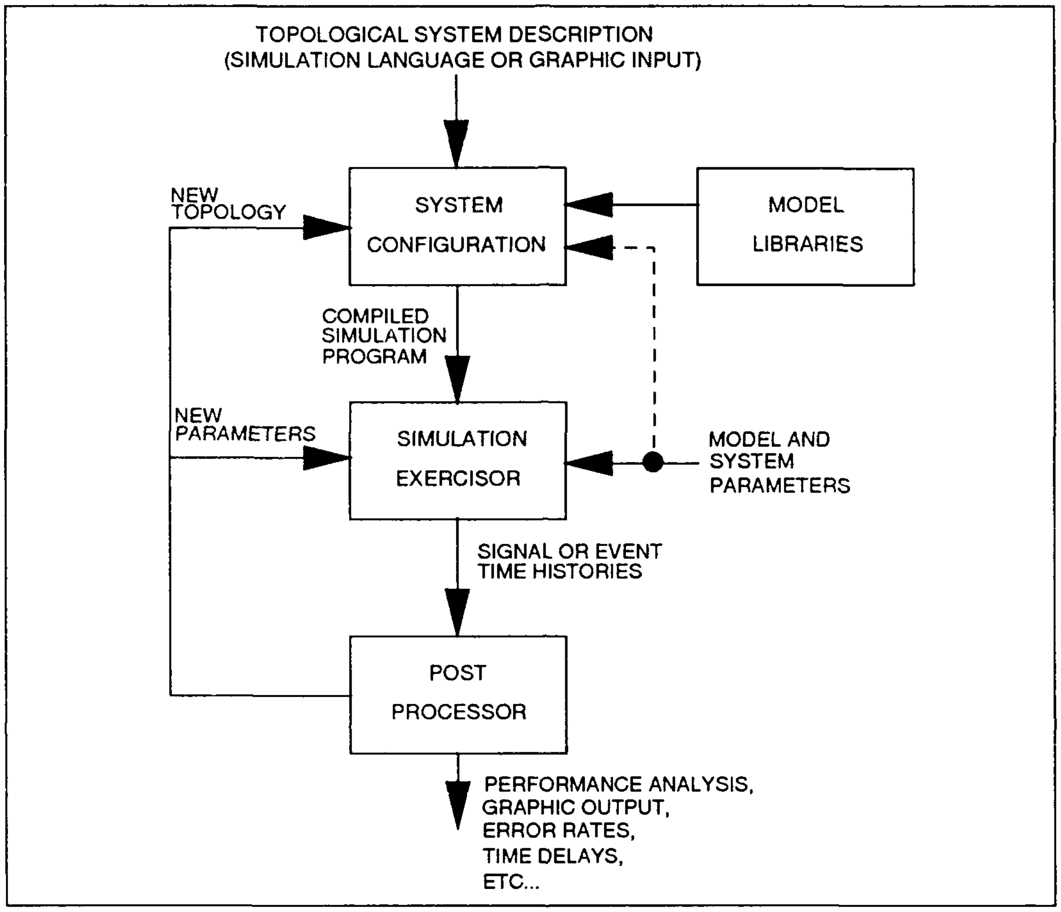
TOPOLOGICAL SYSTEM DESCRIPTION
(SIMULATION LANGUAGE OR GRAPHIC INPUT)
NEW TOPOLOGY
SYSTEM CONFIGURATION
MODEL LIBRARIES
COMPILED SIMULATION PROGRAM
NEW PARAMETERS
SIMULATION EXERCISOR
MODEL AND SYSTEM PARAMETERS
SIGNAL OR EVENT TIME HISTORIES
POST PROCESSOR
PERFORMANCE ANALYSIS, GRAPHIC OUTPUT, ERROR RATES, TIME DELAYS, ETC...

Figure 2.6. Operational Flow of a Simulation Package

2.4.2 Simulation and Programming Language. The software used in the simulator should be written in a higher level programming language such as Fortran, Pascal, or C. These languages, however, do not allow input to the simulator via block diagrams. Therefore, to connect the block diagrams to the simulation exercisor, a preprocessor simulation language is required.

2.4.3 Topological Configuration. The configurator of the simulator software package should allow connection of the model functional blocks in any topological configuration. This feature permits maximum configuration flexibility, however, it may complicate the simulator software structure.

2.4.4 Model Library. The usefulness of a simulation software package depends heavily on the number of functional blocks contained in the model library. The model configurator should allow unlimited nesting of the blocks in the model library to permit any subsystem model to be built. The user should be able to write his own model, and either enter it directly into the system model for simulation or enter it into the model library.

2.4.5 Time and Event Driven Simulation. A simulator should be designed so that processing can occur either at every "tick" of the simulation clock or when identified events occur. For maximum flexibility, both options should be made available to the user. Provisions should be made in the simulator so that the user can designate model blocks either as event driven or time driven.

2.4.6 Postprocessor. The postprocessor function of the simulator should enable the designer to view the results of the simulation. The postprocessor should allow the users to draw direct cause-and-effect inferences about the system operation. As a minimum, the postprocessor should perform functions of common test instruments such as spectrum analyzers, and software utilities such as profilers of resource utilization. Statistical analysis, as well as graphics display routines, are also essential for the postprocessor.

2.4.7 User Interface. The simulation package should be user friendly. The documentation provided with the package should be comprehensive and understandable. On-line documentation, such as "help", should be available and noncryptic to the user.

2.5 Block-Oriented Systems Simulator

The Block-Oriented Systems Simulator (BOSS) provides an interactive environment for block oriented system analysis and design(12:1-2). The BOSS software structure is shown in Figure 2.7. BOSS has a rich environment of block modules in the model library. The supplied blocks are easily altered, enabling new blocks to be designed. "Primitive" blocks may be interconnected to build a new block. These new blocks may be added to the model library.

BOSS has a hierarchical method of building systems to be simulated. BOSS performs consistency checking to ensure proper connectivity between all modules and to verify valid input parameters are used.

When the system to be modeled is stored in the simulator, the parameters are entered and the simulation is performed. The post processor displays the data in various formats. The user decides what data is to stored by inserting probes into the model.

BOSS runs under the VMS operating system on a DEC VAX Station or under the UNIX operating system on the SUN-3 workstation(12:1-2). BOSS provides high resolution bit mapped graphics (1024 x 864) and uses both a mouse and the keyboard for inputs.

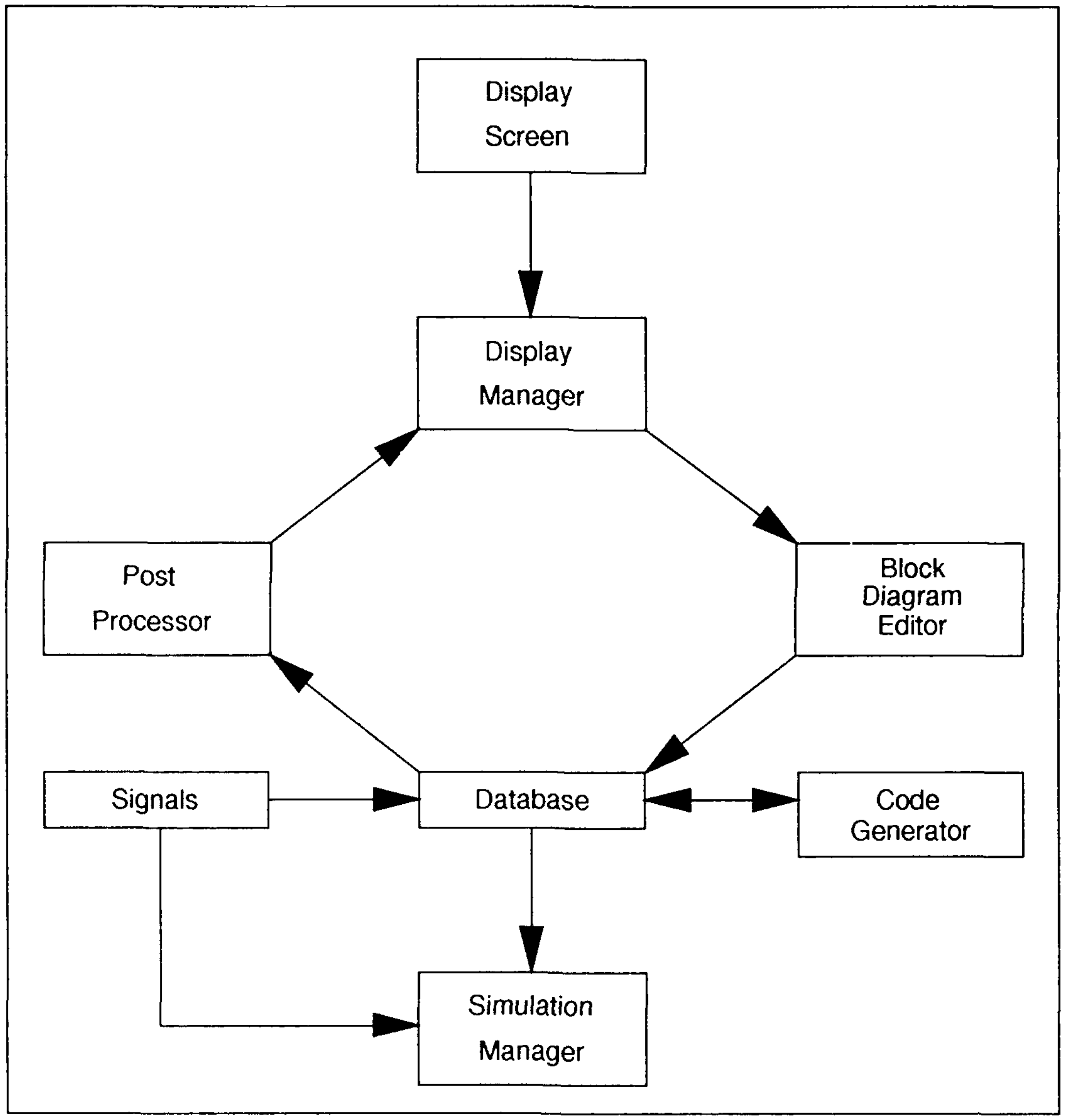

Figure 2.7. BOSS Software Structure (12:1-7)

III. Modeling of the System

The Block Oriented Systems Simulator (BOSS) provides for simulation-based analysis and design of any communication system which can be represented in block diagram form(12:1-2). Simulation models of systems to be evaluated are built using a hierarchical block diagram approach. Once the system to be modeled is expressed in block diagram form, BOSS supports a time-domain, or waveform level simulation(12:1-2).

To build a system to test and verify the Monte Carlo technique and Importance Sampling using BOSS, it was decided to use a simple, real-valued Binary Phase Shift Key (BPSK) system operating at baseband. Built-in BOSS modules were used in the system where possible. Changes made to existing BOSS modules were kept to a minimum. BOSS modules, systems, or parameters will be capitalized to set them apart from generic terms. This chapter will begin with building the basic BPSK system and will then evaluate the system using the conventional Monte Carlo method.

3.1 BPSK System Decomposition

The first step in the BOSS modeling process is to perform a top-down decomposition of the system. The blocks required for a BPSK system are the data generator, the BPSK modulator, the channel, and the BPSK demodulator. Added to the model will be an error detector to identify errors and provide the estimated BER. The next sections will discuss the modeling of these blocks.

3.1.1 Data Generator. BOSS provides a digital data generator in the model library. The generator in BOSS is called RANDOM DATA and is located under the group DIGITAL SOURCES. Figure 3.1 shows the internal modules that make up the data generator module.

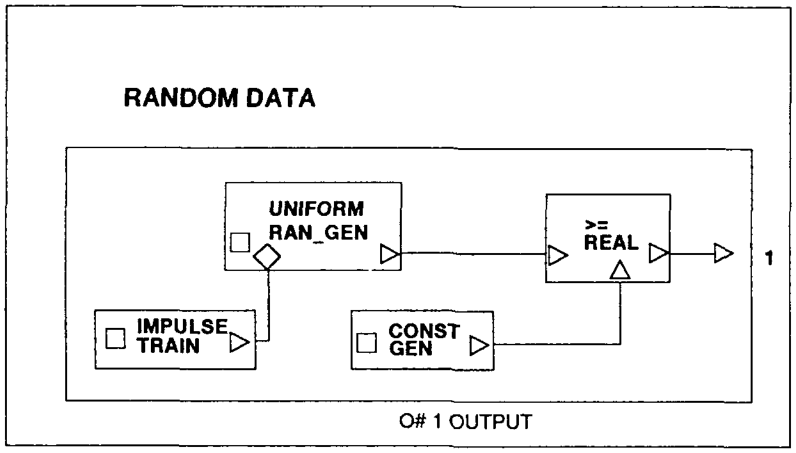

Figure 3.1. RANDOM DATA Internal Modules

RANDOM DATA generates a random binary bit stream (logical "true" or "false") at a specified bit rate and probability of false. RANDOM DATA requires the parameters shown in Table 3.1 to generate the bit stream.

Table 3.1. RANDOM DATA Parameters

ISEED
PROBABILITY OF FALSE
BIT RATE

The parameter ISEED is the seed of the underlying random number generator. The generator has a uniform distribution. ISEED will be renamed DATA GENERATOR SEED to differentiate between the different seeds in the model. This parameter will be passed up to the system level because the DATA GENERATOR SEED will be one of the parameters varied to determine the estimation error. The parameter PROBABILITY OF FALSE will be set internally in the RANDOM DATA module to be 0.5. This setting ensures random data is output from the module. The parameter BIT RATE will be renamed SYMBOL RATE and will be passed up to the system model.

3.1.2 BPSK Modulator. BOSS provides a BSPK modulator in the model library. The modulator in BOSS is called BPSK MOD and is located under the group DIGITAL MODULATORS. Figure 3.2 shows the internal modules that make up the modulator module.

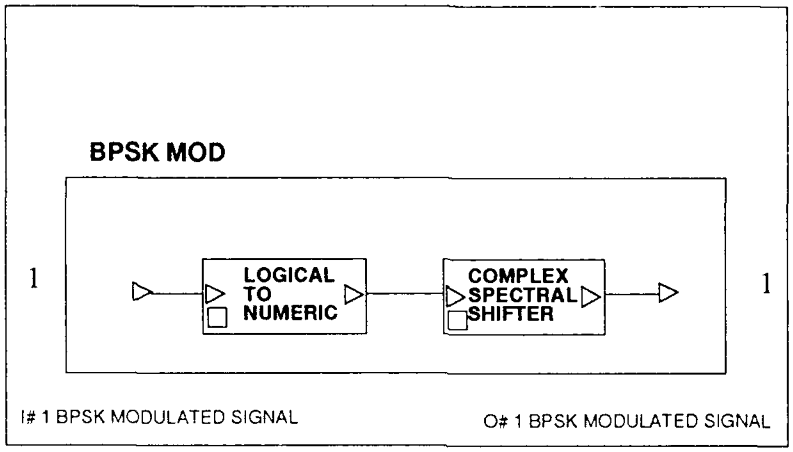

Figure 3.2. BPSK MOD Internal Modules

The BOSS module BPSK MOD modulates the input bit stream to the constellation of (1,0) and (-1,0). In order to set the bit error rate to a desired value later, the values of the constellation were changed to be (TRUE VALUE,0) and (FALSE VALUE,0). Both constellation values were set to be real numbers. Also, because the simulation will be run at baseband, the COMPLEX SPECTRAL SHIFTER block was deleted. To differentiate the user generated module from the BOSS module, the new module is saved as BPSK MODULATOR in the group MISCELLANEOUS. BPSK MODULATOR is shown in Figure 3.3. This module output is connected to both the channel input and to the uncorrupted input of the BPSK demodulator. BPSK MODULATOR requires the parameters shown in Table 3.2 to generate the modulated bit stream.

The parameters TRUE VALUE and FALSE VALUE set the amplitudes of the logical data generator outputs. FALSE VALUE was set internally in module to be

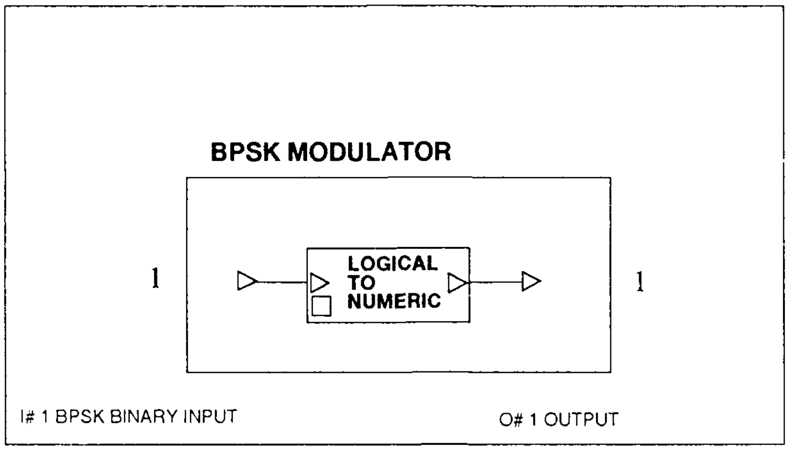

Figure 3.3. BPSK MODULATOR Internal Module

Table 3.2. BPSK MODULATOR Parameters

TRUE VALUE
FALSE VALUE

the negative of TRUE VALUE. TRUE VALUE was passed up to the system level and will be used to set up a predicted BER.

3.1.3 Channel. BOSS does not provide a channel that was desired for this model. The channel for this model requires a real-valued white noise generator and an adder. BOSS provides a generator called GAUSSIAN RAN_GEN under the group NOISE AND INTERFERENCE. The BOSS module GAUSSIAN RAN_GEN generates real-valued Gaussian white noise with adjustable mean and variance. Table 3.3 shows the parameters required to generated the Gaussian noise.

Table 3.3. GAUSSIAN RAN_GEN Parameters

ISEED
MEAN
VARIANCE

The parameter ISEED was renamed NOISE GENERATOR SEED. VARIANCE and NOISE GENERATOR SEED were passed up to the syste.n level. NOISE GENERATOR SEED must be a large odd integer and will be varied to determine the estimation error. VARIANCE will be set to a value of one for conventional Monte Carlo and will be changed for the Importance Sampling module. MEAN was set to 0.0 and stored in the module.

To model the channel, a module called CHANNEL was generated. This module will input tne BPSK signal, add Gaussian noise to the signal, and output the corr ,ted BPSK signal. Figure 3.4 shows the internal modules of the user generated CHANNEL. CHANNEL parameters are the same as GAUSSIAN RAN_GEN parameters shown in Table 3.3.

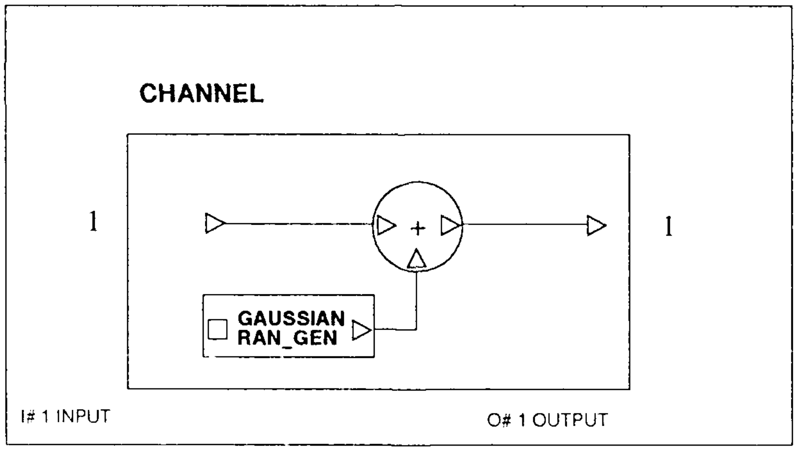

Figure 3.4. CHANNEL Internal Modules

3.1.4 BPSK Demodulator. BOSS provides a PSK demodulator and error counter in the model library. This module is called PSK_DEMOD_AND ERROR_COUNTER and is located under the group DEMODULATORS. PSK_DEMOD_AND ERROR_COUNTER internal modules are shown in Figure 3.5. However, the module supplied by BOSS is configured for complex signals and was changed to work with real signals. The module was further changed to move the error counter outside the module so that changes to incorporate Importance Sampling would be easier. The error counter changes are discussed in the next section.

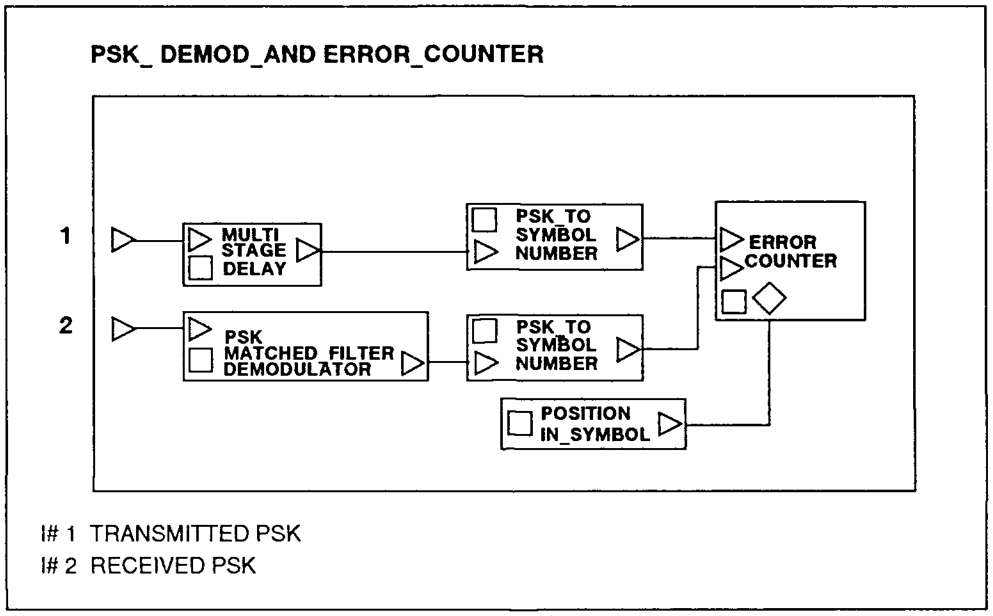

Figure 3.5. PSK_DEMOD_AND ERROR_COUNTER Internal Modules

The BOSS module PSK MATCHED_FILTER DEMODULATOR internal modules are shown in Figure 3.6. The integrate and dump integrator integrates over one symbol interval and dumps the output at the end of each symbol. The PSK DETECTOR quantizes the output of the integrator to the nearest constellation point. In other words, for BPSK, the PSK decides which constellation point the received symbol represents by checking whether the integrator output is greater than or less than zero. Removing the IMAG OF COMPLEX and REAL OF COMPLEX modules, one INTEGRATE AND_DUMP module, and adding a constant generator were the changes necessary for the demodulator to function for real-valued signals. The constant generator was required for the MAKE COMPLEX module input because the PSK DETECTOR requires a complex signal. The modified module was saved as REAL BPSK MATCHED FILTER DEMODULATOR in the group MISCELLANEOUS and is shown in Figure 3.7.

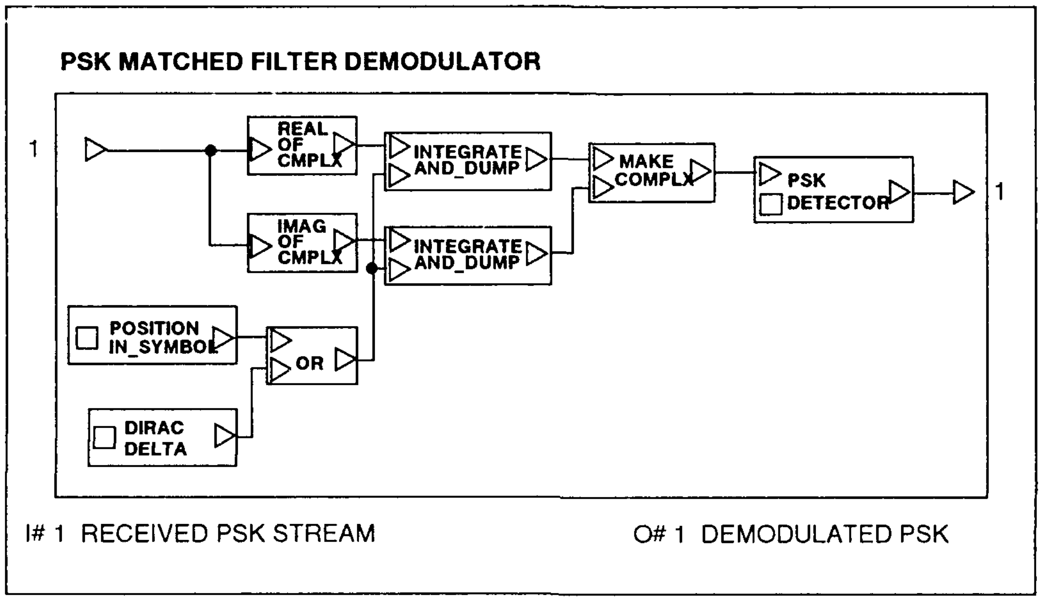

Figure 3.6. PSK MATCHED_FILTER DEMODULATOR Internal Modules

The REAL BPSK MATCHED FILTER DEMODULATOR module requires the parameters shown in Table 3.4. The MAKE COMPLEX IMAG INPUT was set to zero and stored in the module. Because the system is real-valued and channel phase rotation does not occur, CHANNEL PHASE ROTATION (DEG) was set to zero and stored in the module. PSK CONSTELLATION FIRST ANGLE (DEG) was set to 180 degrees and PSK MODULATION ORDER was set to 2, both for BPSK, and stored in the module. The model has no delay elements between data generation and the input to the demodulator, therefore TIME DELAY TO INPUT (SEC) was set to zero. SYMBOL RATE was the only parameter carried up to the system level.

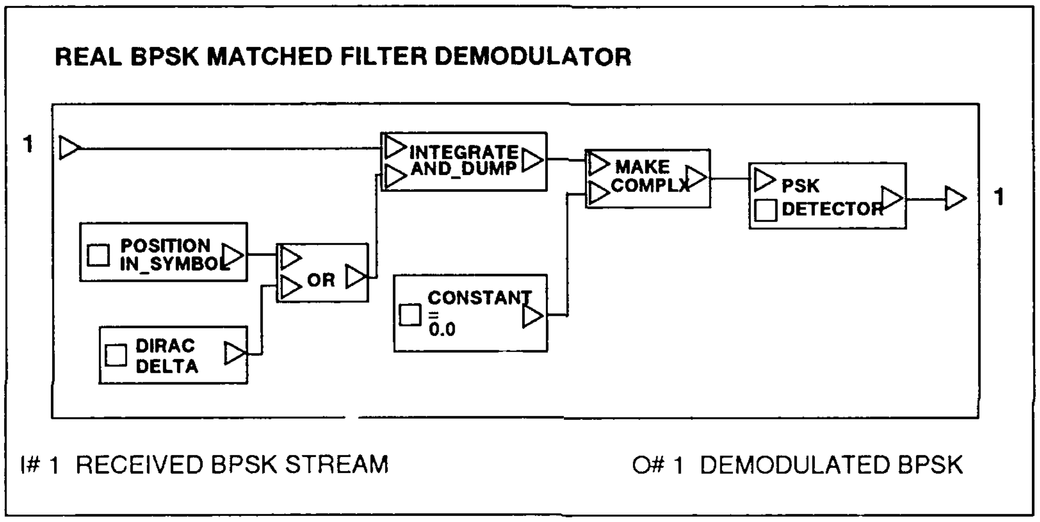

Figure 3.7. REAL BPSK MATCHED FILTER DEMODULATOR Internal Modules

Table 3.4. REAL BPSK MATCHED FILTER DEMODULATOR Parameters

MAKE COMPLEX IMAG INPUT
CHANNEL PHASE ROTATION (DEG)
PSK CONSTELLATION FIRST ANGLE (DEG)
SYMBOL RATE
TIME DELAY TO INPUT (SEC)
PSK MODULATION ORDER

The changes were made to PSK_DEMOD_AND ERROR_COUNTER and stored as BPSK DEMODULATOR under the group MISCELLANEOUS. The BPSK DEMODULATOR module is shown in Figure 3.8. Again, the only BPSK DEMODULATOR parameter to be passed to the system level is SYMBOL RATE.

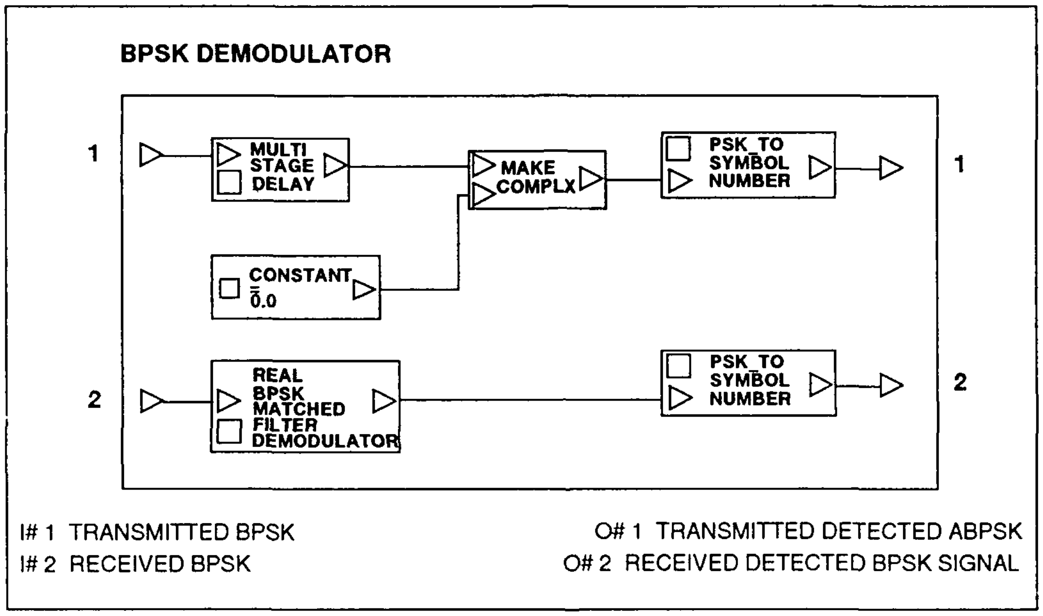

Figure 3.8. BPSK DEMODULATOR Internal Modules

3.1.5 Error Detector. BOSS provides an error detector in the library. The error detector is located under the group DEMODULATORS - INTERNALS. Added to the BOSS supplied error detector was a print module so the output data would include the BER calculation. Also added was a block to stop the simulation when a certain number of symbols were counted. The simulation stopper module was stored as SIM STOPPER and is shown in Figure 3.9. The altered error counter was stored as REAL ERROR COUNTER and is shown in Figure 3.10. This module requires an enable which is added as shown in Figure 3.11 and was stored as BPSK ERROR

COUNTER. The enable is required so the error counter will only be active for the sample at the end of the symbol rather than active for every sample.

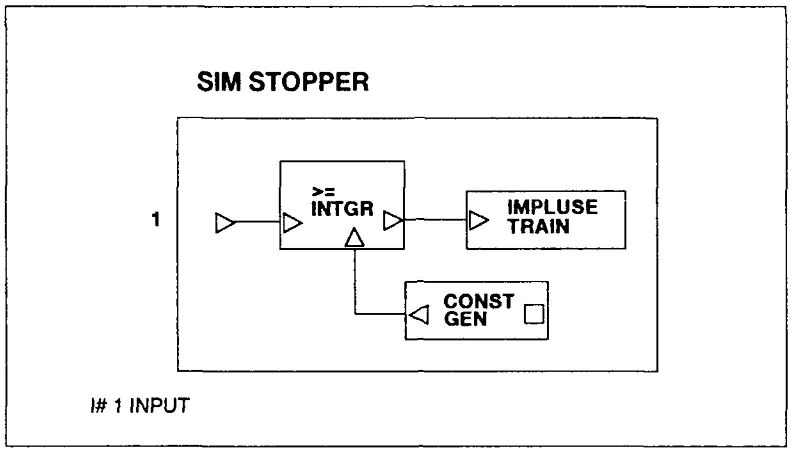

Figure 3.9. SIM STOPPER Internal Modules

The BPSK ERROR COUNTER module counts the number of times its two inputs are unequal. The module prints the number of errors, the calculated BER, and the number of symbols. BPSK ERROR COUNTER requires the parameters shown in Table 3.5.

Table 3.5. BPSK ERROR COUNTER Parameters

PRINT ESTIMATE MODULO
SIM STOPPER NUMBER OF SYMBOLS
BIT RATE
SYMBOL FOR SAMPLE TIME
TIME DELAY TO INPUT (SEC)

TIME DELAY TO INPUT is the number of seconds of delay between transmission of the first bit and that bit appearing at the input of the demodulator. TIME DELAY TO INPUT was set to 10^{-3} seconds because the integrate and dump filter delays the signal by one symbol duration. PRINT ESTIMATE MODULE is an input that determines after how many samples the BER, number of symbols, and number

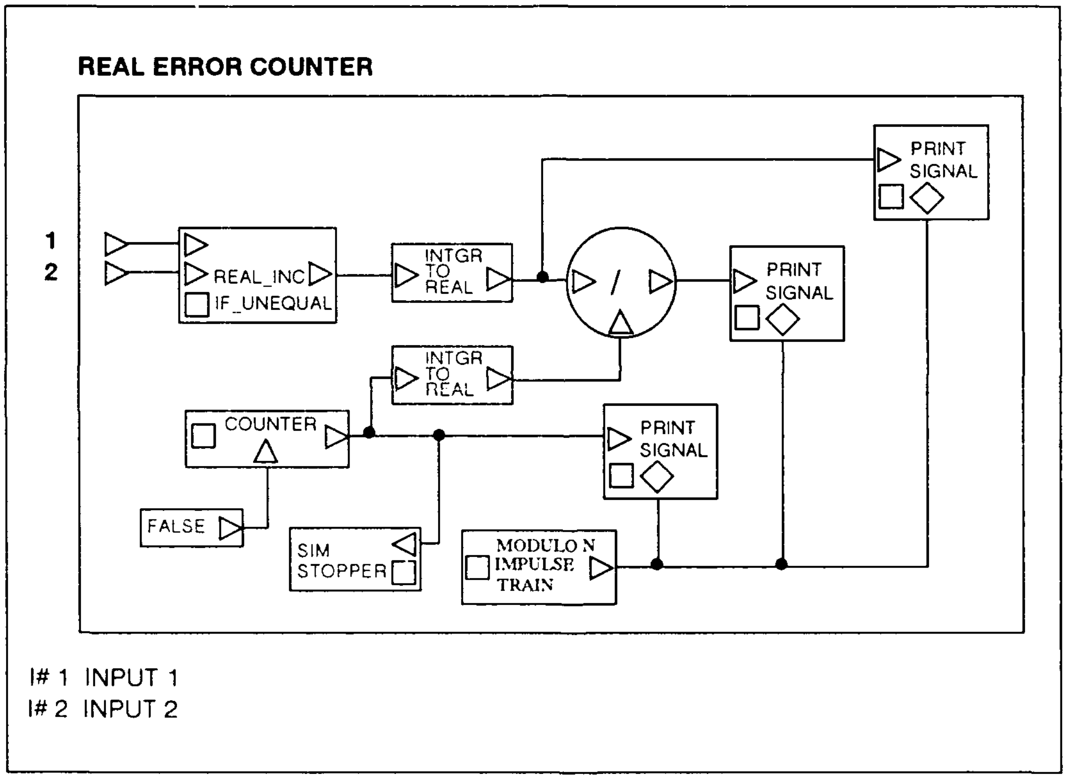

Figure 3.10. REAL ERROR COUNTER Internal Modules

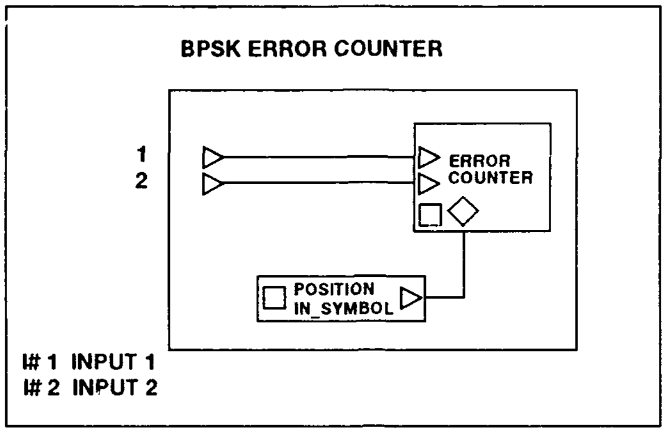

Figure 3.11. BPSK ERROR COUNTER Internal Modules

of bits are calculated and saved. SIM STOPPER NUMBER OF ERRORS is an input that terminates the simulation when a certain number of symbols are counted. PRINT ESTIMATE MODULE and SIM STOPPER NUMBER OF ERRORS were passed up to the system level.

3.2 BPSK System

The modules described in Section 3.1 are connected to form a BPSK system model. This model was named BPSK SYSTEM and stored in the group SYSTEM. Figure3.12 shows the interconnecting of the modules. Table 3.6 shows the parameters required by BPSK SYSTEM.

The parameter STOP-TIME specifies the maximum value of time (in seconds) for the simulation. DT is the time between discrete simulation signal samples (in seconds). The other parameters have been presented in previous sections.

3.2.1 Determining BPSK System Internal Parameters. Values for the parameters listed in Table 3.6 remain to be determined. STOP-TIME is not important

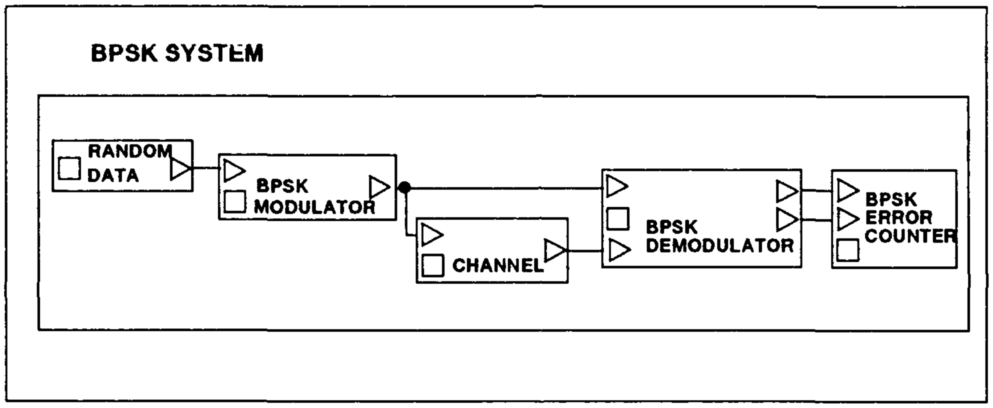

Figure 3.12. BPSK SYSTEM Internal Modules

Table 3.6. BPSK SYSTEM Parameters

STOP-TIME
DT
SIM STOPPER NUMBER OF SYMBOLS
PRINT ESTIMATE MODULO
SYMBOL RATE (HZ)
VARIANCE
NOISE GENERATOR SEED
TRUE VALUE
DATA GENERATOR SEED

to this system because the simulations will be stopped by the symbol count, not by the simulation clock. Therefore, the value of STOP-TIME was set to 10^{10} seconds to ensure the simulator will run until enough symbols are counted. SYMBOL RATE for this simulation is arbitrary, so a symbol rate of 1000Hz will be used for all simulations. DT is the time between discrete simulation signal samples (in seconds). As explained in Section 2.1.1, the signal will be sampled at a rate of ten times the symbol rate, or 10^{-4} seconds. SIM STOPPER NUMBER OF SYMBOLS and PRINT ESTIMATE MODULO are parameters that will vary for different simulations and will be determined later. NOISE GENERATOR SEED and DATA GENERATOR SEED will be the values shown in Table 3.7. Parameter values for VARIANCE and TRUE VALUE determine the expected system BER and will be developed in the next section.

Table 3.7. NOISE GENERATOR SEED and DATA GENERATOR SEED Values

RUN	NOISE GENERATOR SEED	DATA GENERATOR SEED
1	1,599,999,999	1,899,999,999
2	1,699,999,999	1,999,999,999
3	1,799,999,999	2,099,999,999
4	1,899,999,999	1,299,999,999
5	1,999,999,999	1,399,999,999
6	2,099,999,999	1,499,999,999
7	1,299,999,999	1,599,999,999
8	1,399,999,999	1,699,999,999
9	1,299,999,999	1,899,999,999
10	1,199,999,999	1,999,999,999

3.2.2 Determining System BER. To establish baseline parameters for comparing the estimated BER with the calculated BER, the expected system BER had to be determined. The BER for an antipodal BPSK system is calculated using(11:156)

$$BER = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{3.1}$$

where

E_b	*is the energy per transmitted bit*
N_0	*is the white noise power spectral density amplitude*

and $Q(\cdot)$ is the well known complementary error function that is tabulated in many communication texts and is defined by

$$Q(z) = \frac{1}{\sqrt{2\pi}} \int_z^\infty e^{\frac{-t^2}{2}} dt \tag{3.2}$$

3.2.2.1 Determining Energy Per Bit, E_b. Assuming equal energy signals, the energy per bit E_b is(11:157)

$$E_b = \int_0^T s_1^2(t)dt = \int_0^T s_2^2(t)dt \tag{3.3}$$

For BPSK, $s(t)$ is the bit amplitude A. Also, for antipodal BPSK signals, $s_1^2(t) = s_2^2(t) = s^2(t)$. Therefore, the energy per bit E_b is given by

$$E_b = \int_0^T A^2 dt = A^2 T \tag{3.4}$$

3.2.2.2 Determining Noise Factor N_0. The noise factor N_0 is determined by the value of VARIANCE entered and the simulation bandwidth. The simulation bandwidth B_s is a function of the sampling rate f_s, where

$$B_s = 2\left(\frac{f_s}{2}\right) = f_s = \frac{1}{DT} \tag{3.5}$$

and is shown in Figure 3.13. The power, or variance σ_N^2, in a zero-mean, real-valued, Gaussian random variable within the simulation bandwidth B_S is found using

$$\sigma_N^2 = \int_{-f_s/2}^{f_s/2} G_N(f)df = \int_{-f_s/2}^{f_s/2} \left(\frac{N_0}{2}\right) df = \frac{N_0 f_s}{2} = VARIANCE \qquad (3.6)$$

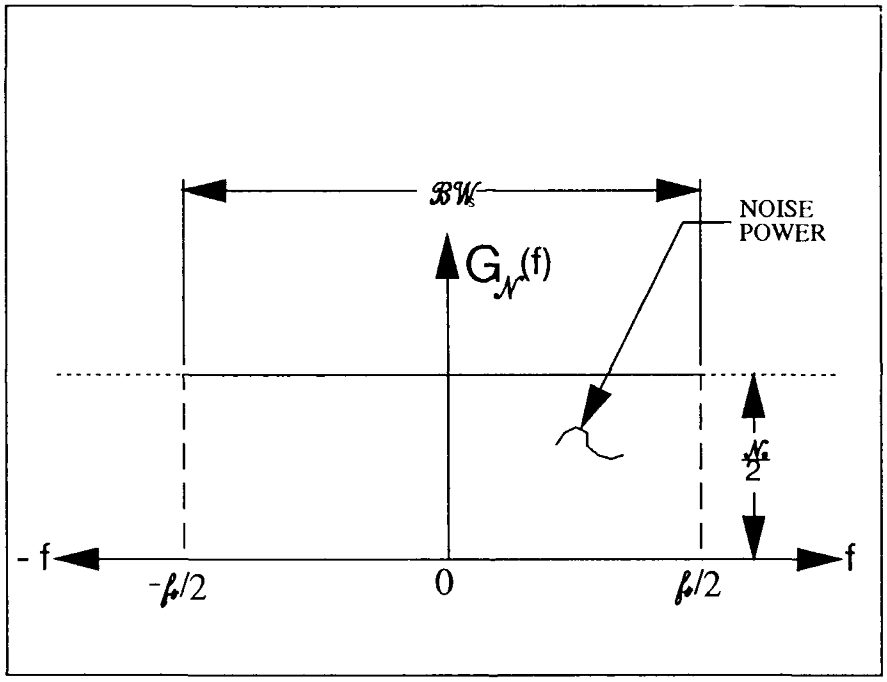

Figure 3.13. Simulation Bandwidth of a Gaussian Random Variable

The value of N_0 is found from Equation 3.6 as

$$N_0 = \frac{2(\text{VARIANCE})}{f_s} = 2(\text{VARIANCE})(\text{DT}) \qquad (3.7)$$

3.2.3 BPSK SYSTEM Simulation Parameter Determination. The BPSK SYSTEM system internal parameters have been determined. Next, the system will be baselined using the conventional Monte Carlo technique. The system parameters remaining to be determined are shown in Table 3.8.

Table 3.8. BOSS SYSTEM Parameters Remaining To Be Determined

SIM STOPPER NUMBER OF ERRORS
PRINT ESTIMATE MODULO
VARIANCE
NOISE GENERATOR IMAGINARY SEED
NOISE GENERATOR REAL SEED
TRUE VALUE
DATA GENERATOR SEED

The performance of the Monte Carlo techniques are to be evaluated by comparing the estimated BER against the calculated BER. A desired calculated BER of 10^{-5} was chosen. This BER value was chosen so the simulation time of the conventional Monte Carlo technique would not be excessive. VARIANCE and TRUE VALUE (the bit amplitudes) determine the system BER. The parameter value VARIANCE was chosen to be one because the the calculations are made easier in the Importance Sampling section. Therefore, the bit amplitudes will be set to determine the system BER. The value for TRUE VALUE remains to be determined.

After combining Equation 3.7, Equation 3.4, Equation 3.1 and simplifying, the BER may be rewritten as

$$BER = Q\left(\sqrt{\frac{(A^2)(T)}{(\text{VARIANCE})(\text{DT})}}\right) \tag{3.8}$$

Solving Equation 3.8 for the bit amplitude A yields

$$A = \sqrt{\frac{(\text{VARIANCE})(\text{DT})}{\text{T}}}\,[Q^{-1}(BER)] \tag{3.9}$$

Substituting in the known parameter values yields

$$A = \sqrt{\frac{(1)(10^{-4})}{10^{-3}}}\,[Q^{-1}(10^{-5})] \approx 1.349$$

3.3 BPSK SYSTEM Verification

The BPSK SYSTEM was simulated using the parameters previously discussed to determine the estimated BER and estimation error. Ten simulations were ran using the seed parameters listed in Table 3.7. The results of the runs are shown in Table 3.9. All simulations took about 2 hours and 54 minutes to run.

Table 3.9. BPSK SYSTEM Using Conventional Monte Carlo Technique Results

RUN	$\hat{P}_e\ (10^{-5})$
1	1.40
2	1.00
3	1.60
4	1.30
5	1.70
6	1.30
7	1.20
8	1.20
9	1.30
10	2.00

The statistics of the BPSK SYSTEM using conventional Monte Carlo techniques are shown in Table 3.10. According to Shanmugan(2:1917), this estimator is "good" because the estimator error ϵ is less than 1.

Table 3.10. BPSK SYSTEM Using Conventional Monte Carlo Technique Statistics

$\hat{P}_e(10^{-5})$	$\sigma^2_{\hat{P}_e}(10^{-11})$	$\sigma_{\hat{P}_e}(10^{-6})$	ϵ
1.40	0.84	2.90	0.29

The next chapter will develop the modules for using the Modified Monte Carlo technique using Importance Sampling.

IV. Implementing Importance Sampling

Most of the BPSK System model developed in the previous chapter is unchanged using the Modified Monte Carlo technique with Importance Sampling. Modules to implement Importance Sampling will be new modules that either replace existing modules or are added to the system model. The new system will be called MODIFIED BPSK SYSTEM. This chapter will begin with building the new modules, incorporating the new modules into the system, and then evaluating the new system as the performance compares with the old system.

4.1 Modified BPSK Syste Decomposition

MODIFIED BPSK SYSTEM requires a biased function generator, a weight generator, and an error counter to calculate the estimated BER. The biasing procedure is implemented by changing the VARIANCE parameter in the module CHANNEL. The unbiasing procedure, however, requires adding a module for calculating the error "weight" used for unbiasing the error count value. An error counter must be added that will accumulate the error weight sum and use that sum for calculating the Importance Sampling estimated BER.

4.1.1 Biased Noise Function. The biasing function $B(x)$ was shown in Section 2.3.1 to be

$$B(x) = \frac{f^*(x)}{f(x)} \tag{4.1}$$

The biased noise function suggested is of the form(3:161)

$$f_N^*(n) = cf_N(n)/[f_N(n)]^\alpha \tag{4.2}$$

where c and α are constants to be calculated so the area under the pdf is one. The equations to determine the constants c and α are shown as Equation2.27 and Equation2.26, respectively.

Equation 4.2 represents the new noise to be added to the BPSK signal in the channel module. Equation 2.26, with $M = 10$ being the number of samples per symbol and $T = 4.3$, yields $\alpha = 0.605$. Equation 2.27 shows that $c = 0.13$. Substituting these values into Equation 4.2, the new noise pdf is

$$f_N^*(n) = 0.25exp(-0.1975x^2) \tag{4.3}$$

This Gaussian pdf has a variance of 2.53.

The previous module CHANNEL did not need to be modified to generate the biased noise function. VARIANCE was again passed up to the system level.

4.1.2 Weight Generator. The weighting, or unbiasing, function is represented by(4:69)

$$w(x) = \frac{f(x)}{f^*(x)} \tag{4.4}$$

This simplifies to(4:69)

$$w(x) = (\sigma_*/\sigma)exp[-(1 - \sigma^2/\sigma_*^2)(x^2/2)] \tag{4.5}$$

Equation 4.5 represents the function that determines the weight of one noise sample within a symbol interval. The module to generate this weight function is shown in Figure 4.1.

WEIGHT GENERATOR requires the parameters shown in Table 4.1. MEAN was set to 0.0 and VARIANCE was set to 1.0. ISEED was renamed NOISE GENERATOR SEED and was passed up to the system level.

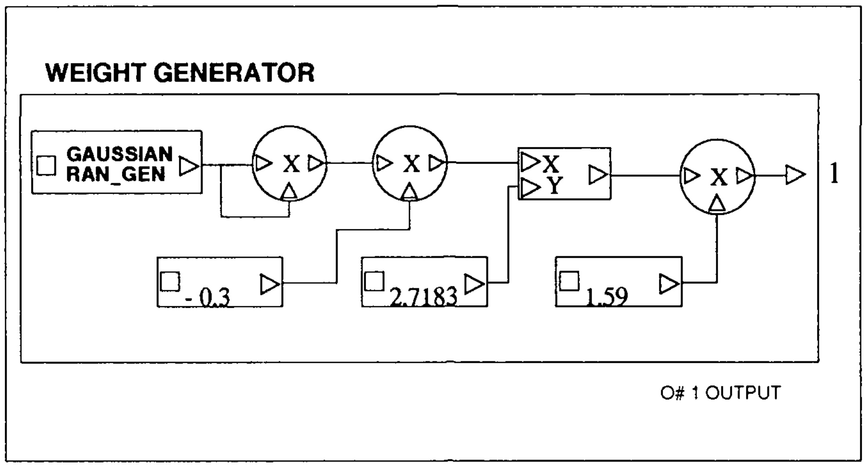

Figure 4.1. WEIGHT GENERATOR Internal Modules

Table 4.1. WEIGHT GENERATOR Parameters

VARIANCE
MEAN
ISEED
VARIANCE
MEAN

This weight generator outputs a value for the noise at one sampling instant. Equation 2.28 shows that the weight of the noise for one symbol interval is the product of the weights of each noise sample in that symbol. WEIGHT FUNCTION GENERATOR, shown in Figure 4.2, takes the value of each noise sample, multiplies the value for 10 samples (one symbol), and outputs the value. Each SAMPLE &_HOLD module samples the value at the symbol fraction shown in the input to each SAMPLE &_HOLD. The TEN INPUT MULTIPLIER multiplies the inputs and outputs the product after 10 samples have been taken.

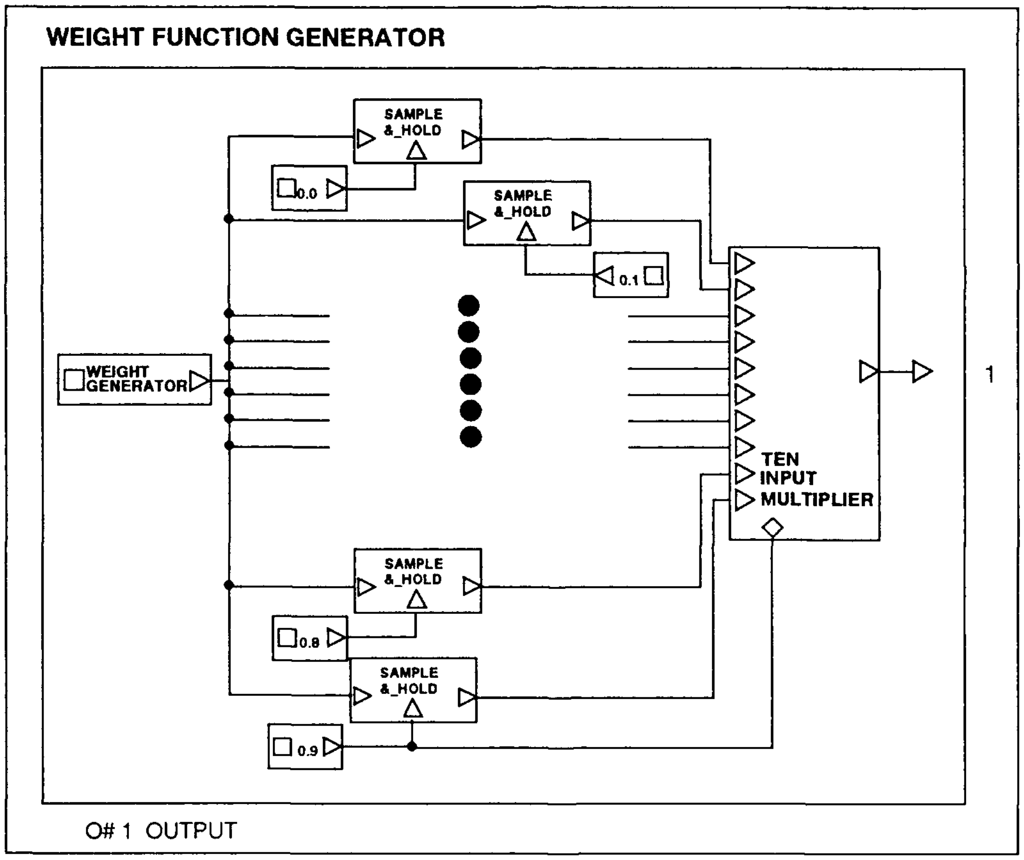

Figure 4.2. WEIGHT FUNCTION GENERATOR Internal Modules

WEIGHT FUNCTION GENERATOR requires the parameters shown in Table 4.2. NOISE GENERATOR SEED will again be passed up to the system level to be varied to determine the estimation error. VARIANCE and SYMBOL RATE (HZ) will be passed to the system level. TIME DELAY TO INPUT (SEC) will be set to 0.0 because there is no delay elements between the sample generation and the input to the SAMPLE &_HOLD module. SYMBOL FRAC FOR SAMPLE TIME is the parameter that determines which noise sample within a symbol each SAMPLE &_HOLD module samples. The SYMBOL FRAC FOR SAMPLE TIME parameters are entered for each module that inputs to each SAMPLE &_HOLD MODULE. These values range from 0.0 to 0.9 in increments of 0.1 and are shown for each module on Figure 4.2.

Table 4.2. WEIGHT FUNCTION GENERATOR Parameters

NOISE GENERATOR SEED
VARIANCE
TIME DELAY TO INPUT (SEC)
SYMBOL FRAC FOR SAMPLE TIME
SYMBOL RATE (HZ)

4.1.3 Importance Sampling Error Counter. Once the weight for the noise in each sample has been determined, the error counter must sum the values of the weights when each error occurs. The estimated BER is determined by dividing the weight sum by the number of transmitted symbols.

Figure 4.3 shows the internal odules of the error counter that determines the estimated BER using Importance Sampling. The module =REAL compares the two inputs and determines if they are equal within a parameter called ALLOWED ROUNDOFF FRACTION. If the inputs are equal, this module outputs 'true', and if the inputs are unequal, this module outputs 'false'. LOGICAL TO NUMERIC converts the true or false outputs from the compare module into numerical outputs. The output is set to 0.0 when the inputs are equal, and 1.0 when the inputs are unequal. The multiplier module multiplies the error weight by 1.0 when an error has occured and the RUNNING SUM module adds and stores the error weight sum.

That sum is divided by the number of transmitted symbols to yield the estimated BER.

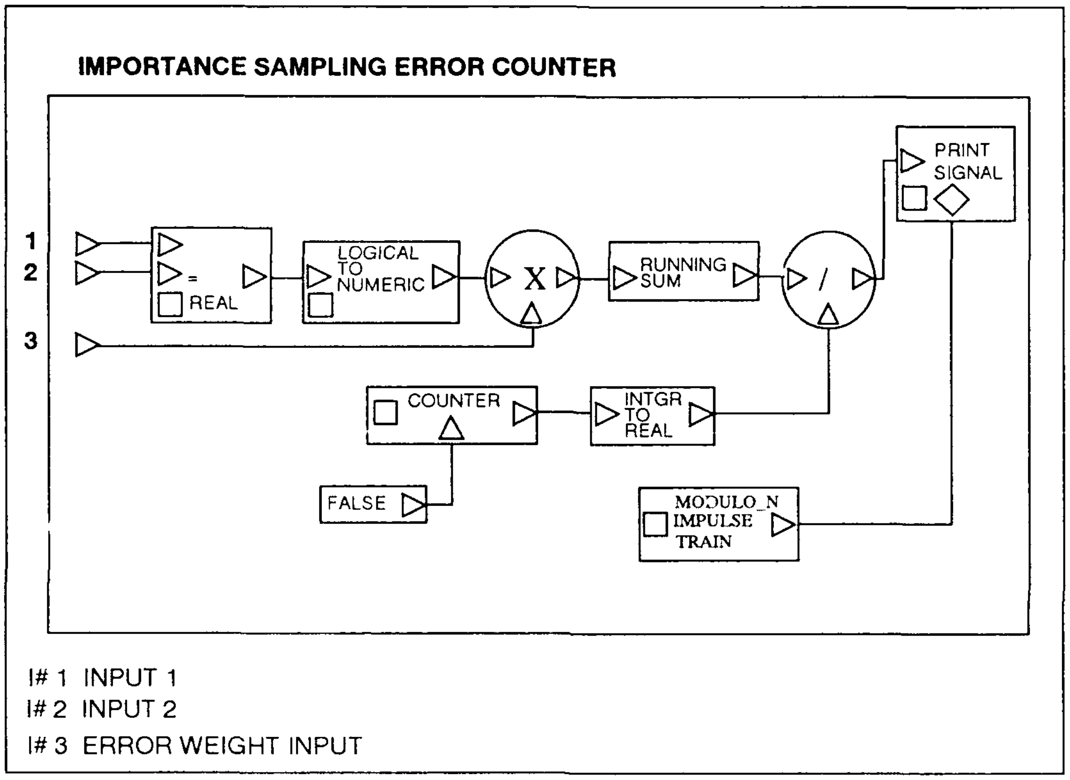

Figure 4.3. IMPORTANCE SAMPLING ERROR COUNTER Internal Modules

IMPORTANCE SAMPLING ERROR COUNTER requires the parameters shown in Table 4.3. ALLOWED ROUNDOFF FRACTION is set to 0.1. TRUE VALUE is set to 0.0 because when the input is true means that an error has not occured. The value of the noise during a symbol not is error is not of interest and multiplying that error weight by 0.0 does not include that value in the running sum. PRINT SIGNAL TEXT was set to read 'Probability of Importance Sampling BER Is: '. PRINT ESTIMATE MODULO was passed up to the system level.

4.2 Modified BPSK System

The BPSK system developed in Section 3.2 was modified to include the modules developed in Section 4.1.3, Section 4.1.2, and Section 4.1.1. The model was named MODIFIED BPSK SYSTEM and stored in the group SYSTEM. Figure 4.4 shows the interconnecting of the modules. Table 4.4 shows the parameters required by MODIFIED BPSK SYSTEM. All of the parameters listed have been covered in previous sections.

4.3 MODIFIED BPSK SYSTEM Testing

MODIFIED BPSK SYSTEM was simulated using the parameters shown in Table4.5. NOISE GENERATOR SEED and DATA GENERATOR SEED were varied as shown in Table3.7.

Table4.6 shows the results of the tests. The results are obviously not what were expected. The model has been extensively reviewed, rebuilt, and tested. Good estimated BER values were never given by the MODIFIED BPSK SYSTEM. Chapter 5 will speculate why the system does not work and suggest some further testing that could be performed.

Table 4.3. IMPORTANCE SAMPLING ERROR COUNTER Parameters

ALLOWED ROUNDOFF FRACTION
TRUE VALUE
FALSE VALUE
PRINT ESTIMATE MODULO
PRINT SIGNAL TEXT

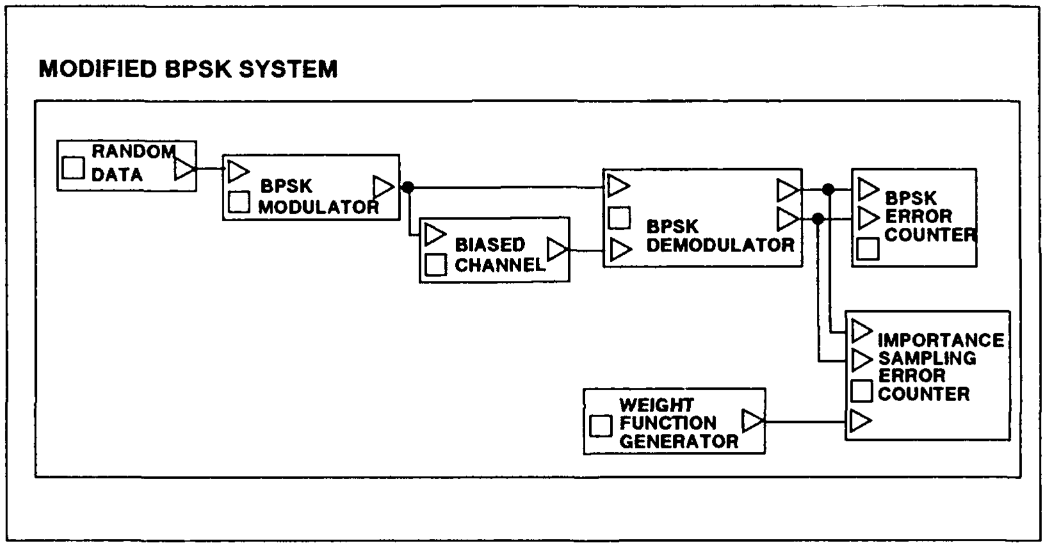

Figure 4.4. MODIFIED BPSK SYSTEM Internal Modules

Table 4.4. MODIFIED BPSK SYSTEM Parameters

STOP-TIME
DT
SIM STOPPER NUMBER OF SYMBOLS
PRINT ESTIMATE MODULO
SYMBOL RATE (HZ)
VARIANCE
NOISE GENERATOR SEED
TRUE VALUE
DATA GENERATOR SEED

Table 4.5. MODIFIED BPSK SYSTEM Test Parameters

STOP-TIME	$1e10$
DT	$1e-4$
SIM STOPPER NUMBER OF SYMBOLS	100000
PRINT ESTIMATE MODULO	10000
SYMBOL RATE (HZ)	1000
NOISE GENERATOR SEED	**Varied**
TRUE VALUE	1.349
DATA GENERATOR SEED	**Varied**

Table 4.6. MODIFIED BPSK SYSTEM Test Results

SYMBOLS	$\hat{\mathbf{P}}_{\mathbf{e}}(\mathbf{10^{-5}})$	ϵ
4000	9.1	8.3
8000	16.9	15.0
24000	29.3	20.0
40000	35.7	16.0

V. Conclusions

This chapter will discuss the test results of the system using conventional Monte Carlo technique and the system using modified Monte Carlo technique with Importance Sampling. Further, possible reasons for the modified system not working are given.

5.1 Conventional Monte Carlo Technique

BPSK SYSTEM was tested using the modules and parameters as explained in Chapter 3. The results of the test are shown in Table 3.9 and Table 3.10. The average of the estimated BER is $1.4X10^{-5}$. The mean was well within the Monte Carlo expected value of $0.5\text{X}10^{-5}$ to $2.0\text{X}10^{-5}$. Further, the estimator error statistic ϵ was 0.29. Again, this was well within the range of 0 to 1, which shows the conventional Monte Carlo technique is a good estimator of the actual BER.

However, conventional Monte Carlo technique requires the simulator to generate 10^{k+1} symbols for a BER of 10^{-k}. For simulations of low probability events, the time for conventional Monte Carlo techniques become excessive. An extension of the conventional Monte Carlo technique is to use Importance Sampling. Importance Sampling is a method of reducing the number of required symbols while retaining the confidence levels of conventional Monte Carlo. The results and discussion of implementing Importance Sampling follow in the next section.

5.2 Modified Monte Carlo Technique Using Importance Sampling

MODIFIED BPSK SYSTEM was built to include Importance Sampling by modifying the BPSK SYSTEM model. The MODIFIED BPSK SYSTEM model and parameters are explained in Chapter 4. Some test results are shown in Table 4.6.

The test results received using MODIFIED BPSK SYSTEM were not the results expected. The estimated BER started out higher than expected, and then gradually got worse as the simulation ran. The estimated BER should have settled into a value in the same range as the conventional Monte Carlo technique did.

The model MODIFIED BPSK SYSTEM was built using the theory of Modified Monte Carlo simulation with Importance Sampling as explained in articles by Balaban and Shanmugan[2], Jeruchim[3], Lu and Yao[4], and Mitchel[6]. The articles, while for the most part similar, do differ in their approaches and methods to setting up an Importance Sampling model.

The only differences between the BPSK SYSTEM model and the MODIFIED BPSK SYSTEM model are the CHANNEL module, the WEIGHT FUNCTION GENERATOR module, and the IMPORTANCE SAMPLING ERROR COUNTER module. Because the BPSK SYSTEM model worked as expected, the unchanged modules kept for the *MODIFIED BPSK MODULE were not considered* to be causing the faulty data. Each of the changed modules will now be discussed as they may have contributed to the error.

5.2.1 CHANNEL Module. There was no actual changes to the CHANNEL module. The change was increasing the parameter VARIANCE to make errors occur more frequently. The fact that increasing the noise power caused errors to occur more frequently was evident in the data. The value to set the VARIANCE parameter to was not clear in the articles. One article optimized the VARIANCE to 2.5 while another article stated the optimum new VARIANCE was about 4.3^2 times the baseline VARIANCE. VARIANCES in the range of 2.0 to 20 in increments of 0.2 were used with no improvement of the data shown.

5.2.2 WEIGHT FUNCTION GENERATOR Module. The weight functions were given in the various articles as either:

1. $w(x) = f_X(x)/f^*_X(x)$, or
2. $w(x) = [f_X(x)]^\alpha/c$, or
3. $w(x) = (\frac{\sigma_*}{\sigma})\exp[-\frac{1}{2}(\frac{1}{\sigma^2} - \frac{1}{\sigma_*^2}x^2]$.

which are all equivalent equations. This weighting function is realized in the WEIGHT GENERATOR module. The only question in this function is from where the input variable x is derived. The articles infer that the variable x is input from the noise sample from the biased Gaussian pdf. However, the WEIGHT FUNCTION GENERATOR output using that input seems to have an amplitude too high for the noise sample.

The estimated BER is determined by

$$\hat{p} = \frac{\eta}{N}$$

where η is the sum of the individual error weights for each symbol and N is the total number of bits counted in the simulation. Using conventional Monte Carlo, the BER is determined by dividing a relatively small number of errors by a relatively large number of bits. In Chapter 3, on the average, each test had 14 errors for 10^6 symbols. Using Modified Monte Carlo, for the estimated BER $\hat{p}$ to remain within the expected Monte Carlo range of values, the number of bits N would decrease by the "sample saving". That would mean that the sum of the error weights η would decrease by the same magnitude. As an example using the conventional Monte Carlo numbers given above, if the sample savings is a magnitude of 10^3, then

$$\sum_{e=1}^{N} W_e \approx 10^{-3}$$

where W represents the sum of the error weights and e is the index for each error.

For all simulations, the weight of each noise sample ranged from 10^{-2} to 10^6 with the average noise sample value being approximately 10^3. This was a discrepancy that can only be explained by the input variable x of the weight function generator being incorrect.

The sample and hold modules that pick out the noise value at a particular sample time in the symbol worked correctly. The multiply module used to form the product of the 10 sample noise weights functioned correctly. The module WEIGHT GENERATOR was tested using a constant generator as the source, rather than the WEIGHT GENERATOR, and the output was as expected.

5.2.3 IMPORTANCE SAMPLING ERROR COUNTER Module. This module was generated from the module REAL ERROR COUNTER. The only changes were for this module to sum the error weights and only output the estimated BER for Importance Sampling. This module functioned correctly when tested as a stand alone module with known inputs. The only critical part of this module is that the error occurrence multiplier arrive at one input of multiply module at the same time the error weight of the noise that caused the error arrived at the other input. The timing was confirmed as correct.

5.3 Probable Problem

From the analysis performed in the previous section, the problem must lie in how the weight of each noise sample is determined. Either the weight function equation is incorrect or the variable x to put into the weight equation was not generated in this module. Every reasonable (and some not so reasonable) input and combination of inputs was used for the variable x, all with invalid results.

5.4 Recommendations

The idea of using Modified Monte Carlo simulation technique with Importance Sampling to simulate low probability events seems an invaluable tool for the researcher. The following items are recommended for further study:

1. Determine the problem with the model derived to implement Modified Monte Carlo technique with Importance Sampling. Test the model to determine the sample savings as a function of system memory, BER, and variance of the new noise pdf.
2. Apply the working model to simulations performed for previous thesis and determine the time saved and samples reduced using the Modified Monte Carlo technique with Importance Sampling.
3. Generate a self-contained module that when inserted into most simulation models only requires the parameters of that particular model be input. In other words, make the Importance Sampling model as "exportable" as possible.

Bibliography

1. Balaban, Philip and K. Sam Shanmugan. "Computer-Aided Modeling, Analysis, and Design of Communication Systems: Introduction and Issue Overview," *IEEE Journal on Selected Areas in Communcications, 2:* 1-7 (January 1984).
2. Balaban, Philip and K. Sam Shanmugan. "A Modified Monte-Carlo Simulation Technique for the Evaluation of Error Rate in Digital Communication Systems," *IEEE Transactions on Communications, 28:* 1916-1924 (November 1980).
3. Jeruchim, Michel C. "Techniques for Estimating the Bit Error Rate in the Simulation of Digital Communication Systems," *IEEE Journal on Selected Areas in Communications, 1:* 153-170 (January 1984).
4. Lu, Dingqing and Kung Yao. "Improved Importance Sampling Technique for Efficient Simulation of Digital Communication Systems," *I EEE Journal on Selected Areas in Communications, 6:* 67-74 (January 1988)
5. Matis, Kurt R. and James W. Modestino. "Interactive Simulation of Digital Communication Systems," *IEEE Journal on Selected Areas in Communications 1:* 51-55 (January 1984).
6. Mitchel, R. L. "Importance Sampling Applied to Simulation of False Alarm Statistics," *IEEE Transactions on Aerospace and Electronic Systems 1:* 15-24 (January 1981).
7. Morgan, Capt Fernando A. *Analysis and Simulation of a Pseudonoise Synchroniation System.* MS Thesis AFIT/GE/ENG/88D-31. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1988.
8. Mouftah, Hussein T. and Charles H. Sauer. "Guest Editorial Computer-Aided Modeling, Analysis, and Design of Communication Networks: Introduction and Issue Overview," *IEEE Journal on Selected Areas in Communications, 1:* 126-129 (January 1988).
9. Palmer, Lary C. "Computer Modeling and Simulation of Communications Satellite Channels," *IEEE Journal on Selected Areas in Communications 1:* 89-101 (January 1984).
10. Shamugan, K. Sam. "An Update on Software Packages for Simulation of Communication Systems (Links)," *IEEE Journal on Selected Areas in Communications 1:* 5-10 (January 1988).
11. Sklar, Bernard. *Digital Communications.* Englewood Cliffs, New Jersey: Prentice Hall, 1988.
12. ST*AR Corporation, Lawrence, KA. *BOSS (Block-Oriented Software Simulator) User's Manual* (Boss version; st*ar 2.02 Edition), 1989.

13. Stoops, Capt Stephen Alan. *Simulation of Electronic Warfare Receivers.* MS Thesis AFIT/GE/ENG/89D-53. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1989.

Vita

Captain John B. Bennett was born on 29 August 1952 in Jackson, Mississippi. He graduated from Forest Hill High School in Jackson, Mississippi in 1970. After attending Delta State University in Cleveland, Mississippi for one year, he enlisted in the USAF in June 1970. After basic training, he served for eleven years as a Wideband Radio Technician before attending the University of Arkansas under the Airman Education and Commissioning Program (AECP). He graduated from the University of Arkansas in 1984 with the degree of Bachelor of Science in Electrical Engineering. After graduation, he received a commission in the USAF through Officer's Training School at Lackland AFB, Texas. After commissioning, he served as an Aircraft Nuclear Safety/Compatibility engineer at the Air Force Weapons Laboratory at Kirtland AFB, New Mexico until entering the School of Engineering, Air Force Institute of Technology, in May 1989.

Permanent address: 104 Sunset Strip
Joliet, Illinois 60435

December 1990 Master's Thesis

IMPLEMENTATION OF THE MODIFIED MONTE CARLO
TECHNIQUE USING IMPORTANCE SAMPLING
ON THE BLOCK ORIENTED SYSTEM SIMULATOR

John B. Bennett, Captain, USAF

Air Force Institute of Technology
WPAFB OH 45433-6583

AFIT/GE/ENG/90D-03

Abstract

The purpose of this effort was to implement the Modified Monte Carlo technique using Importance Sampling on the Block Oriented System Simulator (BOSS). In this thesis, computer simulation techniques of communications systems were reviewed. Next, conventional Monte Carlo techniques and Modified Monte Carlo techniques using Importance Sampling were reviewed. Models of Binary Phase Shift Keying (BPSK) systems using both Monte Carlo techniques were implemented and simulated. Reasons for the model using Importance Sampling not working correctly are postulated.

The Monte Carlo technique is a method of ensuring that an inherently infinite procedure, such as determining system bit error rate (BER), can be determined within an appropriate accuracy and a confidence range after a set number of samples. Conventional Monte Carlo requires 10^{k+} samples be generated to determine a BER of 10^{-k}. This number of samples results in an estimated BER in the range of 0.5 to 2.0 of the true BER. The number of samples required using conventional Monte Carlo techniques can result in unacceptable simulation times for low probability events. Importance Sampling is a method of reducing the number of samples required to determine an estimated BER with the same accuracy and confidence as conventional Monte Carlo.

Monte Carlo, modified Monte Carlo, importance sampling, Block Oriented System Simulator, conventional Monte Carlo, simulation, error weighting, sample reduction, probability of error, Monte Carlo, low probability simulation 71

Unclassified Unclassified Unclassified UL